WHOI-88-15

Woods Hole Oceanographic Institution

Surface-Wave Data Acquisition and Dissemination by VHF Packet Radio and Computer Networking

by

M. Briscoe, E. Denton, D. Frye
M. Hunt, E. Montgomery, and R. Payne

April 1988

Technical Report

Funding was provided by the Office of Naval Research through contract Number N00014-86-K-0751 under the University Research Initiative Program.

2

WHOI-88-15

Surface-Wave Data Acquisition and Dissemination by VHF Packet Radio and Computer Networking

by

M. Briscoe, E. Denton, D. Frye,
M. Hunt, E. Montgomery, and R. Payne

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

April 1988

Technical Report

Funding was provided by the Office of Naval Research
through contract Number N00014-86-K-0715 under the
University Research Initiative Program.

Approved for Distribution:

Robert C Beardsley

Robert C. Beardsley, Chairman
Department of Physical Oceanography

88 7 28 019

Contents

List of Figures

1 Abstract

Waverider buoy data are normally transmitted on a 27 MHz analog radio link to a shore station a few miles away, where the buoy data are plotted on a paper strip-chart recorder or logged digitally for later computer processing.

Instead, we have constructed a relay station on Martha's Vineyard island that retransmits the received Waverider data over a digital, 148 MHz packet-radio link to a personal computer in our laboratory on Cape Cod, where the data are edited, processed, spectrally analyzed, and then sent over an Ethernet line to our Institution mainframe computer for archiving. Telephone modem access of a special wave-data file on the mainframe permits unattended data dissemination to the public.

The report describes the entire system, including Waverider buoy mooring hardware, computer programs, and equipment.

The purpose of the project was to learn what difficulties are involved in the automated acquisition and dissemination of telemetered oceanographic data, and to gain experience with packet radio techniques. Although secondary to these purposes, the long-term surface-wave monitoring off the southwest shore of Martha's Vineyard has its own scientific, engineering, and environmental benefits.

2 Overview

2.1 Purpose of Project

Under the University Research Initiative Program at Woods Hole Oceanographic Institution, a general goal of the Telemetry Project is to develop techniques to gather *in situ* data from the ocean and to disseminate them to users in a timely and efficient way. Since there has been very little experience in this task in the oceanographic community, we constructed this project to gain experience.

In particular, we wished to acquire incoming telemetered data, edit them, process them as appropriate, archive the raw data or some reduced form of them, and make the data available to users. In this context, it made no difference what the incoming data might be, since the data were just something to be handled and disseminated. However, since we wished if possible in this demonstation project to have the users of the data be the public - so as to have a broad interest base rather than a restricted audience – we judged that surface wave data from nearby waters might be a good data source. In retrospect, we might have had even more public interest in the data if they had provided real-time access to offshore winds in either popular sailing or wind-surfing areas. Additionally, the surface wave data provided engineering support information to other tests at the wave-buoy mooring site.

In addition to the goal of learning about the problems involved in dealing with real-time data, their processing, and their dissemination, we were at the same time investigating the possible utility of packet radio techniques for the error-free digital transmission of oceanographic data. As in all data acquisition projects, our experience has been that most of the time spent processing data is devoted to the small fraction of bad data, that is, missing data points and incorrect values. The minimization of this problem by using packet radio was attractive.

Consequently, the wave-data project was designed to provide an incoming data stream of surface wave data from a Waverider buoy moored offshore, with part of the telemetry link using packet radio.

2.2 Options, and Selection of Methods

The Waverider buoy (see section 2.0) is a simple to use, reliable device for the measurement of surface wave displacements; it is readily moored in water of modest depths (20 m is very easy, 200 m is quite possible, 2000 m is quite difficult). Its major disadvantages are its analog AM radio link on 27 MHz, which is a modulation scheme and frequency band subject to considerable interference, and its limited telemetry range, which is some 50 km in the very best of conditions, but which is typically 10-20 km in most applications.

The waters in the vicinity of Woods Hole, Massachusetts, (Figure 1) are popular sport-boating and commercial fishing waters; experience with the long-term survivability of moored buoys, even navigational buoys, is not good. However, the Ocean Engineering Department of the Institution has maintained for some years a site called the "Buoy Farm," which is an engineering test site about 16 km southwest of Martha's Vineyard Island, in 42 m deep water. The Buoy Farm is marked with corner buoys carrying lights and radar reflectors, and is marked on navigational charts of the area. Various mooring, instrument, and materials tests are conducted at the Farm on a regular basis, so it is not much effort to get there for deploying small moorings and for inspection trips. As a side benefit, our wave measurements at the Buoy Farm are in direct support of the engineering tests there.

Unfortunately, the Buoy Farm is about 44 km from our laboratories in Woods Hole, so the direct 27 MHz transmissions from the Waverider buoy would not often be received successfully. The closest land to the Buoy Farm, about 18 km away, is the Gay Head section of Martha's Vineyard, which also has the advantage of being 45 m above sea level, so an antenna site there provides the best reception possible of the Waverider signal.

From Martha's Vineyard the immediate options to get the data to Woods Hole, about 26 km away, were by another analog radio link, by telephone line, or by the digital packet-radio link that we wanted to test anyway. The packet radio link was selected. The frequency used (148.450 MHz) was fortuitious, for it allowed the use of an experimental frequency already allocated to Woods Hole Oceanographic Institution for work near Martha's Vineyard, and it allowed the use of inexpensive and easily available amateur radio equipment normally used on the 144-148 MHz band.

The remaining consideration was the relay station on Martha's Vineyard, somewhere in the Gay Head area. The station would have to receive the Waverider signal, demodulate and digitize it for input (as ASCII) to the packet radio system, and transmit the digital data to Woods Hole. The relay station would have to include sufficient intelligence to know when to sample the data and then either transmit it at certain times, or wait for an interrogation request from the base station at Woods Hole. We elected to transmit at certain times, so as to put some stress on the base station. If effect, we were more comfortable with our capability to interrogate from the base station, so we chose the timed transmissions as the method that would teach us the most.

2.3 Overall System Description

Details of the Waverider buoy and its mooring, the relay station on Martha's Vineyard, the base station in Woods Hole, and the data handling procedures are in the following sections. Here, the overall system description is given.

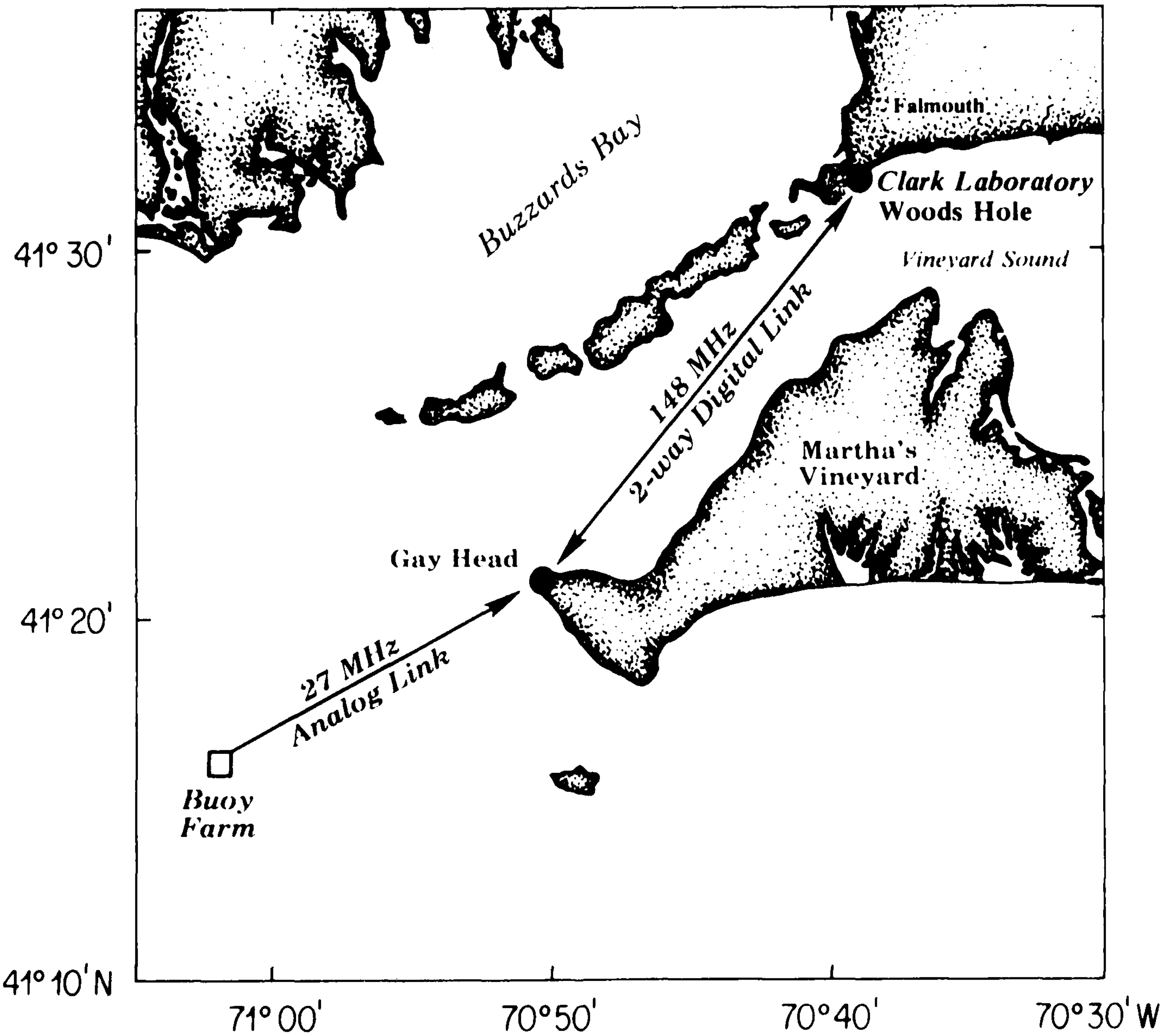


Figure 1: Chart of the area near Woods Hole Massachusetts, showing the location of the moored Waverider buoy at the "Buoy Farm," the relay station on Martha's Vineyard, and the Woods Hole Oceanographic Institution Clark Laboratory in Woods Hole.

Figure 2 shows the overall system. The Waverider is a commercial system that has been modified to include a satellite location-transmitter (ARGOS) in case it should drift away (see section 2.0). The relay station (configured with the identification GAYHD) and the base station (configured as WHOI-1) contain mostly commercial components assembled to our specifications and using our programs. The packet radio part of the system, which includes the ICOM transceivers (see sections 3.0 and 4.0) and the Terminal Node Controllers (TNC), is assembled from standard, unmodified amateur radio equipment.

From Buoy to Martha's Vineyard

The Waverider buoy transmits continuo ısly on 27.595 MHz. The N.B.A. Controls WCR-1 receiving system in the relay station continuously outputs a slowing varying DC signal, -5 to +5 volts, corresponding to -5 to +5 meters of wave displacement. The Elexor PL1000 in the relay station is programmed (see appendix A.1) digitize 2048 12-bit samples (2 samples per second) beginning at 21 minutes and 28 seconds after each hour, and again at 51 minutes and 28 seconds after each hour. That is, 1024 seconds of wave data (17 minutes, 4 seconds) are logged during an interval centered on the even hour and the even half-hour. The timing is based on an internal real-time clock in the PL1000, about which more will be said later. The 12-bit digitizer allows the data to cover the range -4.096 to +4.096 m. The 2048 data points are stored in the PL1000 buffer along with a 2049th word, which is the digitized output from a (nominally) 3.2 volt DC power supply plugged into the main 110VAC power strip in the relay station; this is a monitor of the status of the power to the relay van. Immediately upon completion of this sampling, the PL1000 begins to send formatted ASCII to the MFJ 1270 Terminal Node Controller (TNC).

Data Format for Transmission

The data are formatted as a modification of the drifting buoy (DRIBU) format, used by the World Meteorological Organization and tranmitted routinely over the Global Telecommunications System (GTS). The principal modification is that surface wave data are not part of the DRIBU format, nor of any WMO format. Otherwise, the message is as would normally appear on the GTS; its details are given in Figure 3. The principal advantage of using this modified DRIBU format is that all necessary information (date, time, location, etc.) is included, and is easily extracted without special decoding; the disadvantage is that the format is not an efficient way to transmit the 2049 data points.

Error-Detection

The TNC has a 14 kbyte buffer that will hold the entire wave record plus header and footer. As set up (see Appendix A2), the TNC breaks the record into 128 characters per packet (PACLEN 128).

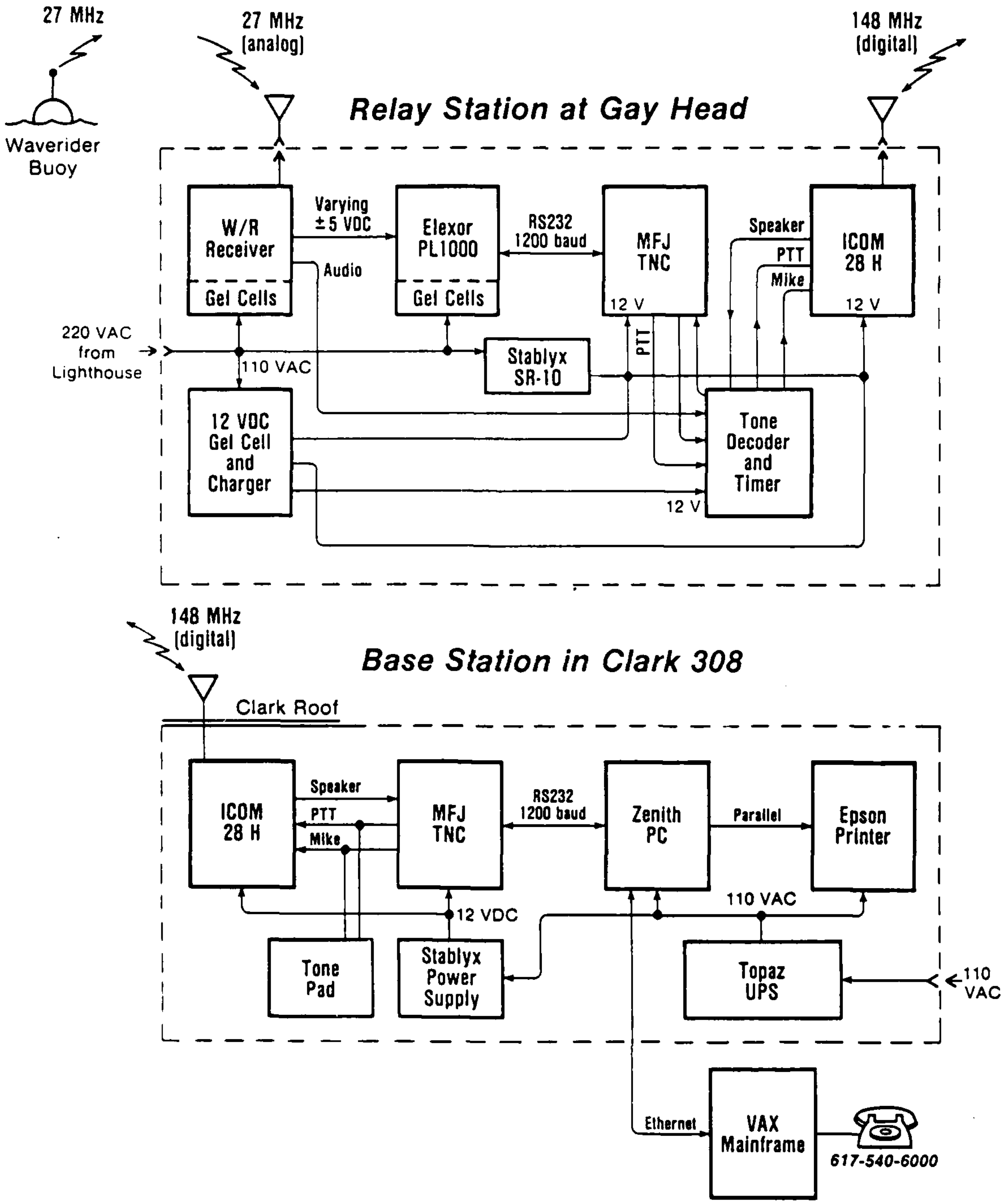


Figure 2: Schematic of the entire system, showing the Waverider buoy, the analog data telemetry, the relay station on Martha's Vineyard, the digital packet-radio link, the base station in Woods Hole, the computer networking of the processed data, and the telephone modem connection for public access to the data.

DATA FORMAT FOR VHF PACKET LINK

```
line no.
  1      ZCZC  nnn
  2      MMNT  KMVY  ddhhmm
  3      ZZXX  ddMMY  hhmm/  74116  07102
  4      999
  5       SWWWW SWWWW SWWWW SWWWW SWWWW SWWWW SWWWW SWWWW
  6       SWWWW SWWWW SWWWW SWWWW SWWWW SWWWW SWWWW SWWWW
  .        .
260       SWWWW SWWWW SWWWW SWWWW SWWWW SWWWW SWWWW SWWWW
261      333  44185  61616  8VVVV 69696=
262      NNNN
```

line no.		
1	ZCZC	starting signal for a GTS msg
	nnn	message sequence number, 000 to 999 (may reset to 000)
2	MM	addressed message (not for global distribution)
	NT	message from North Atlantic area
	K	United States
	MVY	Martha's Vineyard
	ddhhmm	day, hour, minute of transmission of message
3	ZZXX	starting signal for DRIBU format
	ddMMY	day, month, last digit of year of data observation
	hhmm/	hour, minute of data observation; / is block filler
	7	quadrant of data location (North, West)
	4116	41°16'
	07102	71°02'
4	999	starting signal for wave data time series
5	S	+ or -
	WWWW	wave height in millimeters; 8 data words per line
6		same as line 5; 256 data lines
260		same as line 5; 2048 data words total (0.5 s samples)
261	333	starting signal for remaining information
	44	WMO Northwestern Atlantic Ocean area
	185	Waverider ID (buoy serial number is 68185)
	61616	starting group for additional information
	8	indicator for engineering data
	VVVV	remote station voltage indicator (millivolts)
	69696	ending group for additional information
	=	ending indicator for DRIBU format
262	NNNN	ending signal for GTS message

DATA SAMPLING AND TRANSMISSION SCHEDULE

once each minute: WHOI-1 polls GAYHD to check the connect status and GAYHD responds; if no response, WHOI-1 initiates a reconnect

51 min 28 s and 21 min 28 s after each hour: GAYHD samples remote station voltage indicator and begins sampling 2048 data points at 0.5 s interval (takes 17 min 04 s to complete)

08 min 32 s and 38 min 32 s after each hour: GAYHD begins transmision sequence to WHOI-1; data are sent in 128 byte packets (1 to 3 at a time) and each packet or group of packets is acknowledged by WHOI-1 as having been received correctly; total transmision time is about 3 minutes

All starting times are plus/minus one minute.

Figure 3: Data Format for VHF Packet Link

Each packet has a cyclic redundancy check appended, which is recalculated at the base station to see if the packet came through error-free. If so, the base station sends an acknowledgement (ACK) to the relay station and the next packet is sent. If the ACK is not received, the packet is resent until it is received correctly. The TNC in the relay station is set to retry 15 times (RETRY 15) if necessary. The relay station nibbles its way through the data in its buffer until all the data have been sent, which takes about 3 1/2 minutes, waits for the next half-hour's data and starts the cycle again.

Link Failures

It is possible that a power outage or some other problem might interrupt the communications between the relay station and the base station. To test for this condition, the base station sends a check packet once each minute; if the relay station hears the check packet, it sends an ACK. If no ACK is heard, the base will send the check packet up to 15 times, each time waiting for an ACK. Finally, the base will disconnect since it assumes there is no relay station there. However, the base is also set up to attempt to stay permanently connected (CONPERM ON), so it will call GAYHD again and try and establish a connected status. If the link is really down, then WHOI-1 may try forever to contact GAYHD; under the protocol for CONPERM, there is no limit to the number of tries.

A more common error condition is that the base station is not functioning due to (for example) a temporary shutdown for maintainance. If this condition persists long enough, the data transmission from GAYHD will not be ACKed by WHOI-1, and GAYHD will retry out (i.e., it will disconnect) after 15 tries to send a packet. If WHOI-1 comes back on the air before the next half-hour data cycle has sent more data to the relay station's TNC, WHOI-1 will connect again with GAYHD and the data transmission will begin again, possibly without loss of data but more likely with a loss of all the data that had been sent subsequent to the link shutdown. However, if WHOI-1 is off the air for a long time, the PL1000 at the relay station tries to send more data to the TNC, and finally the TNC buffer becomes full and the PL1000 and the TNC become locked up.

System Reset

There are two ways to recover from the lockup described above: (1) simply restarting the base controller program (described in detail in section 5-1 and Appendix B), or (2) doing a remote reset. The resetting of the Base TNC, which occurs at program start-up, will cause the TNC buffer to be emptied and program execution resumed. The data in the buffer will be lost but the following transmission will be received correctly. Alternately, the remote reset is performed by sending standard touch tones from the base to the relay station. The touchtone pair #2, if sent within one second, will cause the relay station's TNC and the PL1000 to undergo a soft reset. That is, both will be reset to their turn-on condition, except for the emptying of

the hardware I/O buffer in the PL1000. The next time the PL1000 clock starts a sampling sequence (at 8 minutes 32 seconds before the next hour or half-hour, whichever comes first) it will fill its software buffer correctly, but the data sent to the TNC will be preceded by the left-over characters in the I/O line, which will be processed as errors by the base station. The next sampling sequence will be handled correctly. Therefore, if the link should be shut down at the base station end, the consequence is a loss of all wave data from the moment of shutdown until the system is reset, including the first record following reset.

Clock Reset

Another system problem, described in more detail in section 4.3 and appendix A.4, occurs when the real-time clock in the PL1000 differs more than one minute from the clock in the base station computer. Neither clock is very good (they both drift 1-2 seconds per day), but at least the base station computer clock can be reset when desired. When the time word in the incoming data stream is detected by the base computer to be more than 1 minute different from the clock in the base computer, a clock reset sequence is sent to the PL1000. In effect, this lets the base computer talk directly to the PL1000 as if the two were hardwired together and the base computer were just a simple terminal. The two radios and TNCs between the base computer and the PL1000 act as a wireless modem pair, and except for the link delays one can talk directly to the PL1000 and even change its internal BASIC program that determines the formatting of the data stream, for example.

Base Station Data Checking

The data stream arriving at the base computer is checked to see if the first few lines (see Figure 3) have the correct format. If not, the entire incoming file is rejected. It is this step that causes the first file following a #2 system reset to be rejected, because the characters left in the PL1000 I/O buffer are not expected by the base computer; this could be programmed around but has been only an annoyance, not a problem.

The base computer determines that the end of file has occured when the characters NNNN are received; 2048 data words are also counted as an additional check. If there were out-of-range values, they are discarded and new values inserted by linear interpolation. This step, which is very important in all our data processing systems based on tape-cassette recording, has not been used so far on even one data point of the approximately 10 million data points so far received. Evidently, the error-detection and send-ACK handshaking protocol of packet radio has done exactly what it is supposed to do.

Data Processing

Section 5.1 describes the data processing. Briefly, power spectra are calculated from the 2048 data points and the spectral coefficients are stored to disk on the VAX that is being used as a file server. Simple moments of the spectrum (e.g., the square root of the area under the spectrum is the root-mean-squared wave displacement, and 4 times this number is defined as the significant wave height) are calculated and stored as well. Although not part of the original project, the 2048 raw data points are sent to another VAX file for the use of Dr. Hans Graber in the Ocean Engineering Department. All communication with the VAX is over a broadband Ethernet line using TCP/IP protocols.

Data Dissemination

Details of the data dissemination scheme used are given in section 5.4. As this was one of the main reasons for this project, considerable effort and priority was given to the planning and implementation of this aspect. Our motivation was the perception that too often the dissemination of telemetered data is not given enough emphasis. In this project, we wished to make the data available through means that were expandable and automated, and which could be used to disseminate data over large distances without special efforts. We chose to use (1) the Ethernet network installed at the Institution, with TCP/IP networking protocols, and (2) data files accessible over dial-up modems. In principle, anyone anywhere could have access to our telemetered data using either the high data-rate networking access or the low data-rate but very simple dial-up access.

2.4 Data Example

During the first six months of operation, the VHF packet system has provided several useful datasets, archived both on the VAX and on floppy disks. There are a number of possible ways to treat the available data, one of which is shown in Figure 3. The plot in Figure 4 shows significant wave height and wave period for the 17th and 18th of December over time. We could have as easily shown simple wave amplitude over time, or the mean and variance of the wave data by week, month and season.

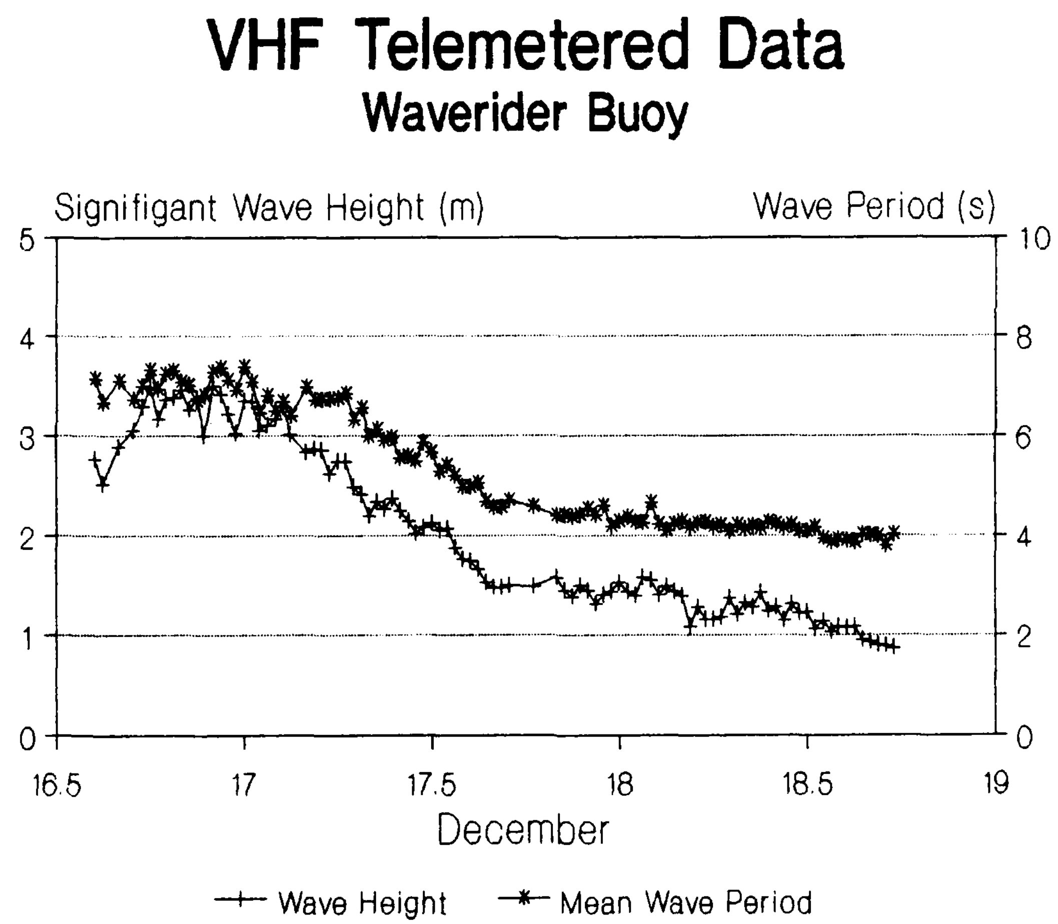


Figure 4: Timeseries of significant waveheight data.

3 Waverider Buoy and Mooring

The Waverider buoy is designed to measure wave height as a function of time in the open ocean, away from structures and other fixed references. It measures wave height by measuring the vertical acceleration of a surface-following buoy using a gravity stabilized accelerometer that is sensitive to accelerations in the 0.035 to 0.65 Hz band. To minimize the effects of horizontal accelerations and buoy pitch and roll on the measurements, the accelerometer is suspended in an oil-filled chamber that is gimballed and damped in such a way as to keep the accelerometer within a few degrees of the vertical and the sensitivity to horizontal accelerations below about 3% (Datawell, 1978). The accelerometer and associated electronics are housed in a 70 cm sphere that floats on and closely follows the surface of the ocean (see Figure 5). To determine wave height from the measured acceleration record, the signal is doubly-integrated to produce the instantaneous water level relative to some mean value. The continuous water-level fluctuations are then converted into a radio signal which is transmitted continuously to the shore-based receiving station at Gay Head on Martha's Vineyard. The Waverider buoy at the Buoy Farm operates on a frequency of 27.595 MHz with a nominal power output of 80 mW. The carrier is amplitude-modulated with a nominally 258 Hz tone; the tone is varied 1.86 Hz per meter of vertical buoy displacement. Accuracy of the wave height measurement is a function of frequency where waves in the important 5-15 second period range are measured within $\pm$ 3% according to the manufacturer's specification.

The standard Waverider buoy has been modified for work at the Buoy Farm by the addition of an ARGOS transmitter, power supply and antenna. The transmitter and battery were mounted inside the instrument well (Figure 6) with a watertight feedthrough used to connect the transmitter to the externally mounted antenna. Transmitter, battery and antenna added about 20 pounds to the Waverider's weight. The purpose of these modifications was to provide a means of monitoring buoy location so that it would be possible to recover the buoy in case it broke free of its mooring. Information concerning Waverider battery voltage and transmitter battery voltage is also being telemetered via the ARGOS system. The mooring design used for the Waverider buoy at the Buoy Farm location is shown in Figure 7. It is designed to allow proper operation of the buoy in currents up to 2 knots and at the same time ensure that the mooring line does not reach the surface under slack water conditions so that it will not get snagged by passing vessels. The small floats attached to the polypropylene line are used to keep the line off the bottom and thereby avoid chafe. A unique feature of most Waverider moorings is the use of a rubber cord beneath the buoy. This very stretchable element (it will stretch to five or six times its original length before it parts) is strong and impervious to seawater, but provides a very soft connection to the lower mooring. This eliminates sharp tugs on the buoy as it follows the wave surface, even when it is stretched out by a strong surface current which would contaminate the acceleration measurement by the introduction of high frequency motions. This type of mooring has been used successfully with Waverider buoys in many environments all over the world.

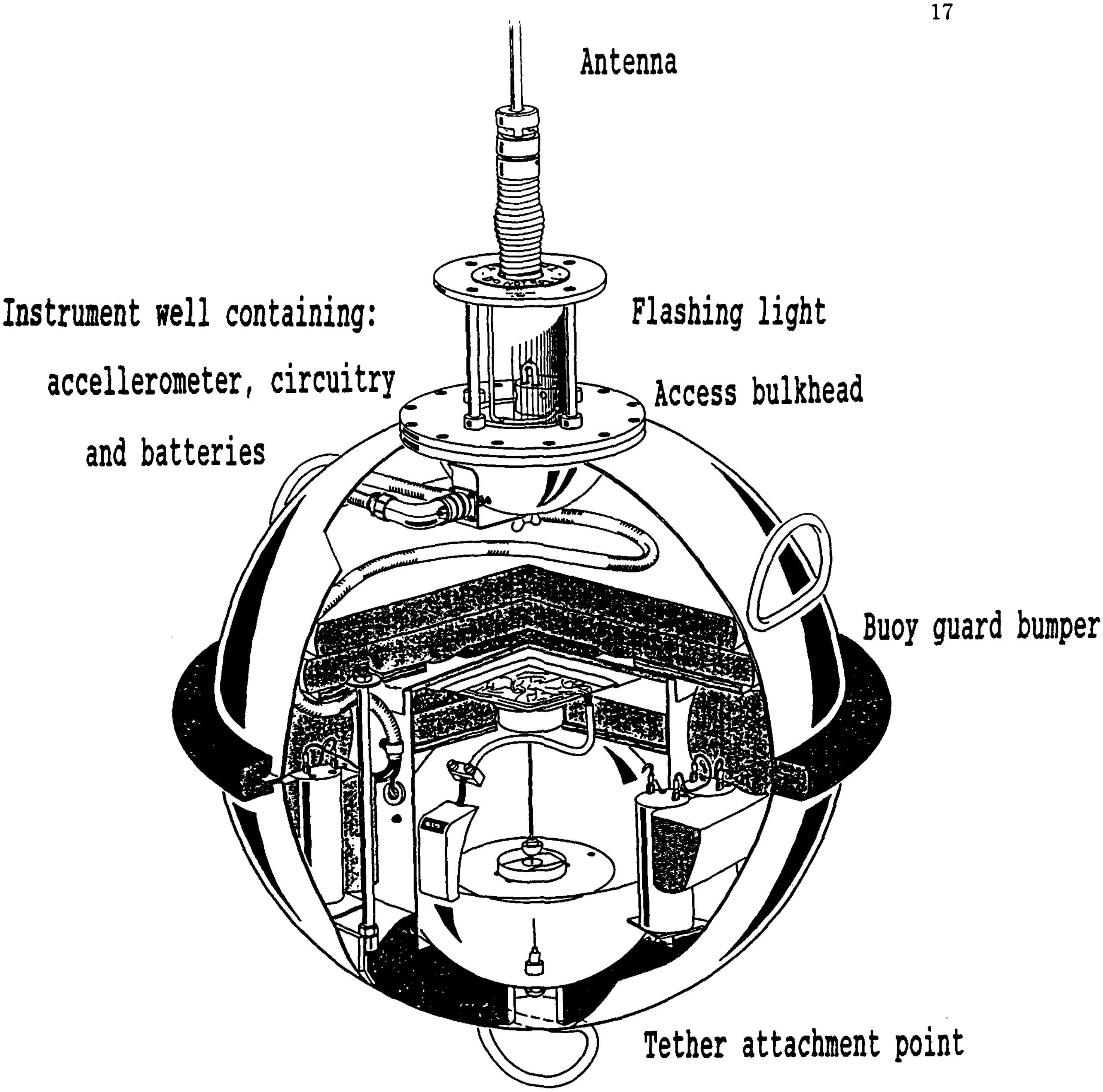


Figure 5: Waverider buoy cut-away [from Datawell brochure].

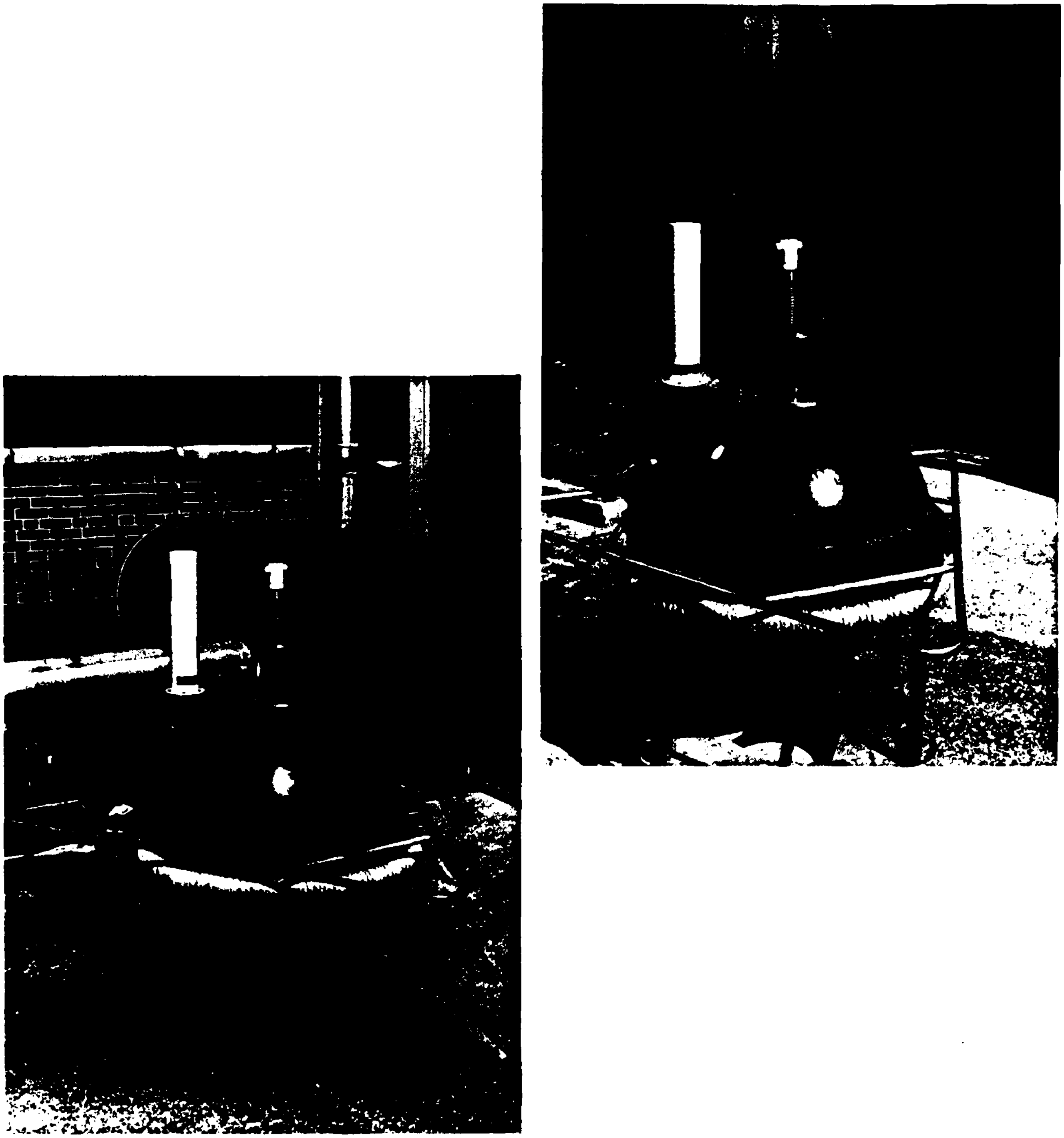


Figure 6: Modified Waverider buoy showing the ARGOS transmitter.

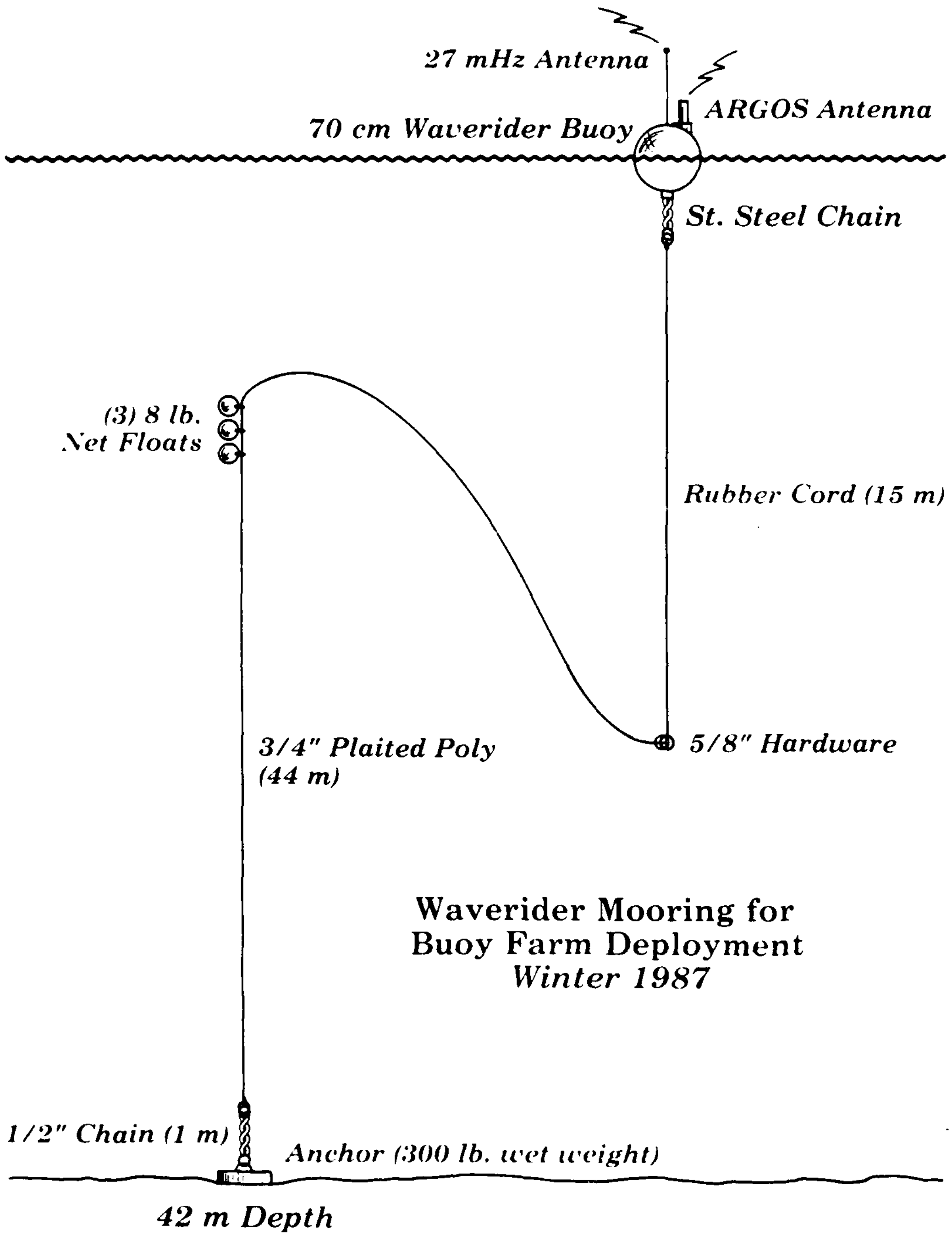


Figure 7: Waverider mooring schematic for Buoy Farm deployment.

4 Relay Station

The relay station on Martha's Vineyard is an essential part of this project because the low-power 27 MHz transmission from the Waverider buoy moored at the Buoy Farm (Figure 1) is not strong enough to reach our laboratories more than 40 km away. In fact, other signals (renegade Citizen's Band operators using high-power equipment and operating on unauthorized frequencies as "HF-ers") illegally using the same frequencies are able to interfere with the Waverider transmissions even when the signal is as strong as it is at Gay Head.

This section describes the equipment at the Gay Head site on Martha's Vineyard, and reviews the various permissions we have obtained to work there. The software used is described in appendix A.

4.1 Description of the Van and Equipment

The military-surplus van being used is a small enclosure originally designed as an electrical shelter to be carried by 3/4 ton trucks; in military usage it might carry portable telephone exchanges or power generators, for example. Entry is through a full-height door at one end. The van is approximately 4 feet wide by 6 feet deep by 5 feet high, is made of aluminum, has high mechanical strength and an ability to withstand exposure to high winds and a salt air environment. The van has skids and lifting pads, which makes it adaptable to oceanographic work ashore or on a ship. Figure 8 shows the van in its installed location near the lighthouse at Gay Head. interior. The van contains the following equipment:

1. Waverider receiver, manufactured by N.B.A. Controls, Ltd., Model WCA-1. It is battery powered, with 110VAC float charging of the battery.

2. A/D Converter and I/O Controller, manufactured by Elexor Associates, Model PL1000. It is AC powered but switches to internal batteries if the AC line fails.

3. Packet Radio Terminal Node Controller (TNC), manufactured by MFJ Enterprises, Model 1270. It requires 12VDC, which is supplied by the main battery in the van if the 110VAC-12VDC supply (see item 4) should fail.

4. VHF Radio Transceiver (25 W), manufactured by ICOM, Model IC-28H. It requires 12VDC, which is provided by a 110VAC-12VDC 10A power supply, Stably Model 10-SR. If the AC line should fail, the main battery in the van will power the transceiver.



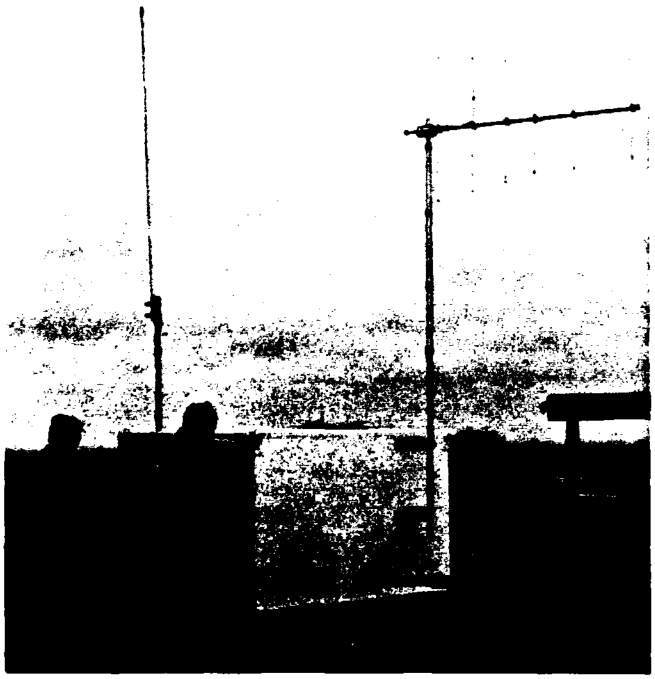


Figure 8: Relay station van next to Gayhead lighthouse on Martha's Vineyard.

5. Tone sequence decoder and timer, WHOI designed and built, based on a Model TSD decoder board from Engineering Consulting.

6. Gel Cell Standby Battery, from Globe, Model GC 12550.

7. Gel Cell Charger, from Globe, Model GRC 14060.

8. 6-Element VHF Yagi Antenna, manufactured by Cushcraft, Model Proline PLC-1426.

9. Vertical Antenna for 27 MHz, Coaxial Half-Wave Type, unknown manufacturer and model.

Power

The primary power for the van is provided by the nearby lighthouse 220VAC supply through Type UF cable, size 12-3. The line runs underground over the ten-foot distance from the lighthouse to the van.

Within the van, a dual-circuit breaker panel provides 110VAC distribution by means of wiremold strips. The 12 V gel cell (item 6 above) is a standby battery in case the AC power should fail; the battery is kept charged by item 7 above, and will supply the TNC (item 3) and the VHF transceiver (item 4) with power if necessary. The Waverider receiver (item 1) and the PL1000 (item 2) have their own internal batteries for standby use.

An inadvertent test of the standby power systems occured when an electrical storm caused the breakers in the van to trip. Approximately 60 hours of backup power is available for all parts of the system except the the Waverider receiver, whose batteries fail after only 24 hours. Ground-fault interrupter breakers were in use at the time; they have now been changed to standard breakers.

4.2 Site Considerations and Permissions

Martha's Vineyard has a considerable number of tourists on it each summer, most of whom visit the southwest corner of the island, which is the part closest to the Waverider site at the Buoy Farm. We felt that an unattended relay station on that part of the island would have even more problems than do our moorings in accessible areas. Conveniently, however, the Coast Guard maintains a lighthouse at Gay Head, so we approached Coast Guard Group Woods Hole to seek permission to operate there. The area of Gay Head was nearly ideal for our purposes because of its height about 45 m above sea level, and because it is nearly line-of-sight from the roof of the Institution's Clark Laboratory in Woods Hole, where our base station was to be located. In addition to height, the location next to the lighthouse had the potential of providing an AC power line to keep our electronics powered. Although

this was not essential – we could have provided other power sources – it kept the cost and complexity down, and greatly increased the reliability of the relay station.

A written request was sent to the Coast Guard (19 December 1986) asking for permission to install the relay station at the Gay Head site. This was followed by a meeting with Group Woods Hole at which the Coast Guard explained to us that the lighthouse property and the responsibility of maintaining the grounds had been contracted over to a local environmental group, the Vineyard Environmental Research Institute, who would be contacted by the Coast Guard on our behalf. We received permission from the Coast Guard on April 15, 1988 and installed the van at the Gay Head lighthouse on April 22, 1987. A Coast Guard Chief helped with the installation, especially with the provision of the AC power line from inside the lighthouse.

Our original permission was for a 6-month deployment of the van at the Gay Head location; this duration was thought to be long enough for us to work out the technical details of the project and to gain the desired experience with the data dissemination. Indeed, that this report can be written is evidence that the technical side of the project has been largely successful within our 6 months anticipated project length. Also, we did not wish to unduly stress the Waverider mooring in winter storms, or to gamble with losing the buoy.

However, we have received several requests to maintain the project through the winter storms season. The researchers concerned with effects of waves on the erosion of shorelines point out that our data set so far is unique, but not yet very useful. The major damage to shorelines like that of Martha's Vineyard occurs during winter storms, especially in February, and these researchers have specifically asked if we can maintain the Waverider buoy at the Buoy Farm and the relay station at Gay Head for the winter season. Subject to equipment needs and an extension of our permission to have the van at Gay Head, we will try to accomodate these requests for additional environmental information on the wave climate off Martha's Vineyard. It costs little to keep the project running, so the public funds being spent on the project will be more effectively applied if it remains in place through the winter.

4.3 Installation and Checkout

The van was installed at the site by using a 3/4 ton pickup with a hydraulic tail gate; the van was simply lowered to the ground and slid off into position. Because of the potential for high winds at the site, the van was spiked to the ground using four-foot lengths of one-inch black iron pipe that had a flange at the top. The pipe was passed through the bottom tie-down rings of the van.

The two antennas were mounted using commercial television "Y" wall brackets backed up with 1/4 inch aluminum plates on the inside of the van. The 6-element VHF yagi was pointed at the Clark Laboratory using a compass bearing.

Mr. T. Simonds, a resident of Martha's Vineyard and an experienced technician on digital and radio-frequency equipment, has acted as our volunteer, remote aide for the few events where an on-site inspection was needed. In fact, the equipment has been remarkably trouble-free; the only major event (described in section 3.1) was a power outage at the van caused by the tripping of a ground-fault interrupter circuit breaker (now replaced by a non-GFI breaker).

5 Base Station

The base station for the VHF Packet Telemetry system is located in the Clark building at the Woods Hole Oceanographic Institution Quisset Campus. This building is tall and located on a hill so provides a good site for radio communication.

This section describes the equipment at the base, and some of operations details. The software controlling operations at the Base is described in appendix B.

5.1 Laboratory Equipment

The base station for the data telemetered from Gay Head is located in Clark 308. The antenna, a 6 element VHF Yagi identical to that on the relay station is mounted on the top of the east end of the roof of the Clark building. From there the antenna has an almost perfect line-of-sight path to Gay Head. The coax cable (Belden 9251 RG-8U) is run down the east stairwell to the third floor, where it is run in the celing to room 308.

The system components located in Clark 308 are the following:

1. VHF radio Transceiver - ICOM model IC-28 H
2. Packet radio terminal node controller - MFJ Enterprises Inc. model MFJ 1270
3. Regulated power supply - Stablyx model 10-SR
4. Tone pad and switch for WHOI designed tone sequence decoder
5. Zenith P.C. with two disc drives - model Z-148-2
6. Uninteruptable power supply - Topaz Powermaster model 84462
7. Epson dot matrix printer - model MX-80
8. Ethernet board and controller software - 3COM model 3C501
9. FTP (File Transfer Protocol) software - PC/TCP
10. WRPROC (data acquisition, analysis, and storage) software - WHOI designed and implemented

Items 9, 10 will be discussed in section 6.

Primary Power

The base station is run on standard 110VAC available in Clark. All the elements of the receiving system are plugged into the uninterruptable power supply, which can provide approximately 20 hours of backup power, in case of an outage in Clark.

5.2 Installation and Checkout

The components are installed as shown in Figure 2. At power up, the TNC board will have the right-most LED illuminated, the transceiver should be on low power on 148.50 MHz, simplex, the volume control dial should be at 9:00 o'clock, and the computer should have completed its start-up checks, and booted successfully. The CONPERM option in the TNC is set to ON, so the packet boards automatically try to connect whenever they are disconnected. When the two stations are successfully connected, the two right-most LEDS on the TNC will be illuminated. The data acquisition program is started by placing the program disk in drive A; and the formatted storage disk in drive B:; typing WRPROC [CR] and entering [CR] again when prompted to do so. The only maintenance needed is changing the floppy in drive B: weekly, and keeping paper in the printer.

5.3 Troubleshooting Link Problems

There are two major kinds of problems that are likely to occur: (1) the Waverider signal is bad, or (2) there is a timing problem between the PL1000 at Gay Head and the Zenith at the receiving station. There are different diagnostic procedures, and different "fixes" for each kind of problem, so they will be discussed separately.

Questionable Waverider signals are usually indicated by the mean waveheight; the normal mean value output shown on the printer is usually about 0.4 . A very low mean indicates that the phase-lock loop was "unlocked", and thus data values of 0009 (9mV) replace real data values in the data stream. If the Waverider signal is suspect, it is checked with the tone pad. The Waverider receiver at Gay Head is accessed and connected to the 148 MHz link by entering the numbers 0751 in less than 4 seconds. Using 0751 on the tone pad, allows one to listen to the audio signal that comes across the link; this is actually a retransmission on the VHF link of the 27 MHz signal received at the relay station. If there is an unusual amount of static or other noise, there may be a problem with the Waverider itself, or it may simply be that there are other signals overwhelming the Waverider signal, for example interference from illegal users of that frequency. The voltage of the Waverider battery is available in the ARGOS output, and should be checked. An abnormal battery voltage could be the source of signal irregularities.

The WRPROC program (see Appendix B-1) starts waiting for telemetered data at 05 and 35 minutes after each hour. If the Zenith clock gets more than about

4 minutes behind the PL1000 clock, than WRPROC starts listening for data <u>after</u> it has started to be transmitted, so the first characters coming in are wrong, and the data set is discarded. Conversely, if the Zenith gets more than about 5 minutes behind the PL1000, then the Zenith <u>stops</u> waiting before the data are transmitted Consequently, it is dangerous to reset the Zenith clock (using the MS-DOS TIME command) without also resetting the PL1000 clock; Appendix A.4 gives the PL1000 remote clock-reset instructions.

6 Data

This section covers the system modules that acquire the data, process it, store it, and make it available to the user community. The data system was set up as a prototype for future operations. The modularity of the design is apparent in Figure 9, a block diagram of the data handling system. Incoming data rates for the Waverider data and other data sets we anticipate in the near future are, for the most part, fairly low because of the limitations of the transmission media. Acquisition and processing can be accomplished readily with inexpensive microcomputers such as the PC clone we are using to acquire and process the Waverider data. File serving, on the other hand, requires a multitasking machine. It should be capable of handling several channels of processed data and communications from several users simultaneously. At present, it is most economical to do this with a general-use time sharing Digital Equipment Corporation VAX computer at WHOI. Data archiving is convenient on the VAX with 9 track tapes and it has connections to a wide variety of communications modes. If our usage should increase in the future to where it was uneconomical to use the VAX, we could substitue a DEC MicroVAX with no change in software or a UNIX machine with some program development.

6.1 Data Acquistion and Processing

Data acquisition and processing are accomplished on a PC clone by the program WRPROC, written in the C language, and a set of subroutines written in C and assembly language. A program report, describing the program in detail with listings of the program and subroutines and examples of output data are included in Appendix B. A general description follows here.

At half hour intervals, a set of 2048 values of the Waverider output digitized at half second intervals is received and passed to the microcomputer. The 17 minute 4 second acquisition period is centered on the hour and half hour. Data communication starts at about $8\frac{1}{2}$ minutes after the hour and half hour. Processing and file transfer is complete by 15 minutes after the hour and half hour.

A power spectrum is computed with a fast Fourier transform (FFT) subroutine. The following parameters are computed from the data and spectral coefficients:

1. Waverider bias = mean of all values
2. Maximum crest = maximum wave height
3. Minimum trough = minimum wave height
4. Mean wave period (normalized first-moment of the spectrum)
5. Zero-upcrossing wave period (normalized second-moment of the spectrum)

Simplified Diagram of the Data Processing System

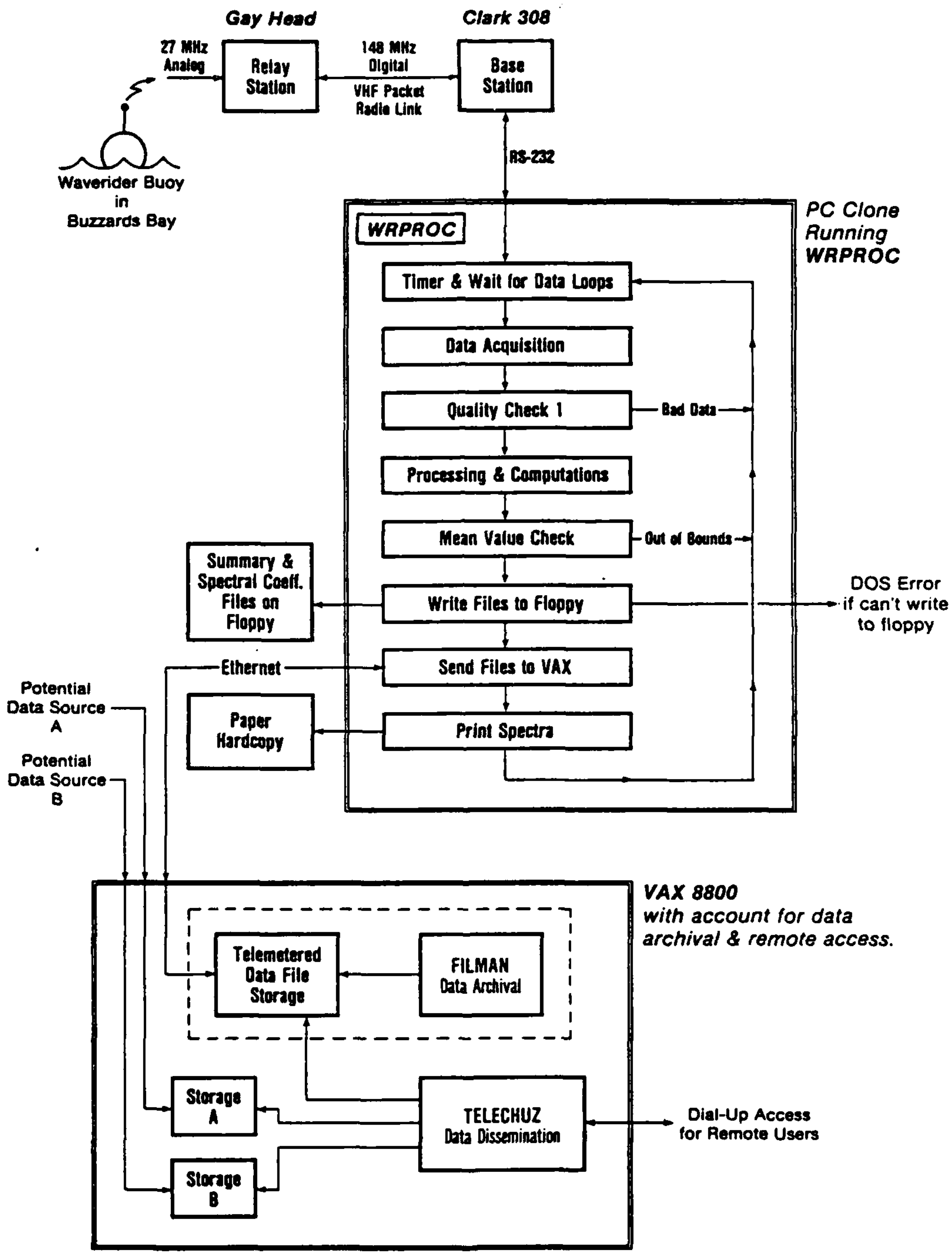


Figure 9: Diagram of the modular data handling system.

6. Total variance of the spectrum (area under the spectrum)
7. Significant wave height (four times the square-root of the variance)
8. Frequency of the iargest spectral peak.

The Waverider bias is subtracted from the individual data values before any of the other computations are made. Details of the analysis can be found in the WRPROC program report (Appendix B1).

Date, time, significant wave height, some operational parameters, and a plot of the spectrum are sent to the printer. See Appendix B.3 for an example. Three files are sent to the VAX via an Ethernet link using FTP TCP/IP software:

1. *Raw data file.* The 2048 wave sea height values.
2. *Spectral coefficient file.* Contains the time, the spectral coefficients, and the moments computed from them.
3. *Computed product file.* Contains date, time, total variance of the spectrum, significant wave height, mean wave period, upcrossing wave period, Waverider bias, maximum crest, minimum trough, and frequency of the spectral peak, at one record per half hour interval.

Examples of the spectral coefficient and computed product files will in Appendix B.2.

6.2 Error and Malfunction Handling

We have designed the system to run unattended and to recover from errors and malfunctions whenever possible. The microcomputer and printer are run from an uninterruptible power supply. There are several error conditions and malfunctions that are addressed by WRPROC:

1. Printer:
   (a) If the printer is not ready when the program is started, an alarm sounds and an error message appears on the screen.
   (b) If the printer paper runs out, an alarm is sounded and the printer output is redirected to the monitor.
2. Garbled incoming data: WRPROC checks the first few records, which do not contain Waverider data, to be sure they have the expected format. If they do not, the whole file is rejected. It is also rejected if an incorrect number of data samples are received. Out-of-range values are replaced with linearly

interpolated values. If the Waverider bias is not within prescribed limits the whole file is rejected. Through the 6 months of operation so far there have been no transmission errors in the data, Although radio frequency interference at the Base can cause the data to become garbled after receipt.

3. If the file server (VAX) is down and cannot accept the data files, WRPROC stores them on its own disk for manual transmission at a later time.

4. If the VHF link transmission time (in the first group of characters sent) differs by more than one minute from the expected transmission time, WRPROC returns a message to the PL1000 causing it to update its internal clock. This keeps the PL1000 and the microcomputer synchronized.

6.3 File Storage and Handling

Some file manipulation is carried out on the file server to make data dissemination and archiving more convenient. A listing and description of the program FILMAN.COM, written in VAX VMS DCL, which carries out the manipulations is presented in Appendix C. We will describe the file management functions for each of the three types of files received by the file server. Each day begins with the 0000 UTC data file. Each week begins at 0000 UTC on Saturday.

1. *Spectral coefficient files*: WRPROC appends the spectral coefficents from the current half hour to the data already accumulated that day and sends the whole updated file each half hour. Thus, by 2350 UTC that day, 48 files exist, only the last one containing all the data from that day. All the files but the last one are deleted (purged). This complete raw data file is kept for another 24 hours to allow any one who wants the data to acquire them. At 2350 UTC the following day, the file is deleted. To make this clearer, at 2350 UTC on May 14 the files from May 14 are purged, leaving one May 14 file, and the file from May 13 is deleted. We do not archive the raw data.

2. *Computed product files*: This is the primary data dissemination file. These files accumulate half hourly in the same manner as the raw data files and are purged at 2350 UTC each day. In addition, weekly files consisting of the concatenation of daily files are generated and archived. At 2350 UTC each day, that day's file, after purging, is appended to the file WRTHSWK.DAT, containing all the data from the current week. At 2350 UTC on Friday several changes are made in file names. The file WRLSTWK.DAT, containing the data from two weeks previous, is renamed WRYRMODA.DAT where YRMODA is the date of the first day of that week. For example, on 29 May the file containing the data from the week beginning 16 May 1987 would be renamed WR870516.DAT. The file WRTHSWK.DAT is renamed WRLSTWK.DAT, and a new file WRTHSWK.DAT is opened for the data from 0000 UTC, 30 May. We begin the week on Saturday so that on Monday morning we can

look at the file WRTHSWK.DAT and see what has happened since we last looked on Friday afternoon.

3. *Raw data files*: These are sent each half hour to an account on the VAX of an investigator who wants the data. We do not archive the original data.

Once a month the daily spectral coefficient files and weekly computed product files are copied to a 9 track tape for archiving. After one month the computed product and spectral coefficient files are backed-up then deleted from the disk.

6.4 Data Dissemination

In order to experiment with making the data easily available through a variety of communications channels, we store it on a disk connected to a VAX. This computer is connected by DECNET to a number of other WHOI computers, to a PACX system which allows other connections; to a high speed cable (WHOI network) which is routed to all the WHOI buildings, and to the telephone system through a multiport modem. The WHOI computer network is connected to an international network system on which most of the major oceanographic institutions also appear as nodes.

The naming of the data files has already been described. Knowing the disk drive and account name PODA:[I33.TELE] one can straightforwardly copy any of the files to another account on any of the WHOI DECNET computers or to any of the computers in the international network. Anyone can call in through the telephone system or the WHOI PACX system and access the data through the program TELECHUZ, described in Appendix D. To connect to this program call 617-540-6000, for 300/1200, baud modems. To the prompts given in lower case letters, respond with the upper case characters: (<CR> means carriage return)

<CR>
enter class: RED <CR>
class red start

Username: I33_TELE <CR>
Password: (can be obtained from the authors)

The user will be guided by a set of menus and prompts. Appendix D.2 contains a sample session, showing the menus and options.

TELECHUZ allows the user to access the Waverider data and other data which become available in the future through a series of menus. At present these menus offer the three data files described above and a choice of getting the data one screen at a time or a whole file at once. A flow diagram of the program and a listing of the program and its subroutines appears in Appendix D. For security purposes the

program and the account are set up so that, upon acceptance by the VAX of the username and password, the user is captive in TELECHUZ. He can move between any of the menu screens within the program but, beyond that, can only log off. The complete file descriptor for each file accessed is given for copying to another account.

6.5 Routine monitoring

Although the data system has been designed to run virtually by itself, there are a few tasks which require human intervention:

1. Daily - Look at the printer output from several half hour intervals to be sure everything in the system is working and sensible data is being processed. Add printer paper when required.

2. Weekly - Check files on VAX disk to be sure all data is getting to the VAX and that the FILMAN.COM program continues to resubmit itself. Insert a new floppy disk in the PC. Check the PC clock time.

3. Monthly - Archive spectral coefficient files on nine track tape. Delete computed product and spectral coefficient files from the disk.

6.6 Lessons

Probably the weakest links thus far have been the analog transmission between the buoy and Gay Head, and the link to the Institution's Digital VAX computers. Our account is on the Red VAX, the Institution's general time sharing computer. The disk drives, however, are connected to the Blue VAX, which is connected to the Red by DECNET. A failure in either the Red or Blue VAX or the disk drive will prevent the PC from getting data to the disk drive. Because the whole day's accumulated Fourier coefficient and computed product file is sent each half hour, it is only the last file of the day which is crucial, but downtime in one or another VAX has required that we move some data from the PC to the VAX manually. The Red VAX was upgraded from an 11/780 to an 8800 during May 1987 and numerous other changes were made in hardware configurations. Downtime connected directly with the swap was small but problems resulting from it have caused considerable downtime. The result is that we cannot rely on the VAX for data archiving but must archive locally at the PC until we are sure the data is on the VAX disk drive.

To keep the program as simple as possible we began with the assumption that the data reaching the PC would be error free. The digital transmission link from Gay Head to the Clark building seems to be completely reliable but the analog data tranmitted from the buoy to Gay Head has suffered from interference. The result in the data has been groups of 8 or more consecutive identical values near zero.

There are two alternatives for responsible data processing in a situation like this:

1. Program around known data problems to recover as much data as possible.
2. Provide a diagnostic parameter which gives a good indication of the data quality.

Since we are recording data much more frequently than is required by the data users, we can afford to lose some data and therefore chose option 2. The diagnostic parameter is the mean value over all 2048 sea height values per half hour observation period. This drifts slowly but stayed generally within the limits of 0.32 to 0.41 during the first six months of operation. Values outside this range were a reliable indication that bad data had been received.

7 Summary

7.1 Unanticipated Problems

As in any developmental technical project, there have been numerous small problems, most of which happened only once or twice until we were able to implement a fix. An example is the previously mentioned power outage at the relay station caused by an electrical storm and a sensitive GFI circuit breaker; this was fixed by replacing the GFI with a standard circuit breaker.

More serious are a set of recurrent problems that we have not been able to eliminate. The most devastating system failure occurs when the Zenith computer at the base station has an error in trying to read one of its floppy disc drives. The base station halts until someone manually resets the Zenith; no data are passed over the link until this is done. We are sure this error is correctable; we have not yet been able to do it. Equally frustrating is a paper jam in the printer logging the wave spectra; the printer will work for days, even weeks, then jam.

A more insideous data loss occurs when there is interference on the 27 MHz link from the Waverider buoy to the relay station at Gay Head. The editing algorithms in the base station data processing package are only able to discard any data that show this interference: the symptom is an unlocked phase-lock loop in the Waverider receiver in the relay station, which yields nominally zero voltage to the PL1000, which comes through the system as a series of approximately four seconds or longer of zero waveheight. When the base station sees this symptom, it flags and discards that data segment. If the fault occurs several times in a 1024-second wave record, the entire record is discarded. This is an especially frustrating problem because it is caused solely by illegal operation on the frequency we are using; much of the operation seems centered in the midwestern part of the country, but this may simply be an accident of the 27 MHz propagation paths to our site.

A final problem, caused by hardware but not yet solved, is the drift of the real-time clock in the Zenith computer. The clock drift in the PL1000 in the relay station would be no problem if the Zenith were correct, because the PL1000 clock is reset from the Zenith's clock. Sections 1.3 and 4.3 describe this problem in more detail. Here we mention only the surprising fact that it seems very difficult to get an accurate time signal easily. The cost of getting a high-quality good time mark (within 1 ms, say) is high. We initially thought we could acquire the system time from our Institution VAX, but it turns out that this is set arbitrarily from a computer operator's personal wristwatch whenever it is needed, with no controls on its accuracy. Digital time marks are available over the radio, but the interfacing equipment is typically costly (several thousand dollars) except for a 10 ms unit from Heathkit; a time mark is also available to 50 ms with a telephone call via a dial-up modem at the Naval Observatory, but this is a toll call and requires the communications packages and equipment. Finally "internet" time is

supposed available over the computer networks that WHOI is part of; we have not yet implemented this possibility.

A different kind of unanticipated problem is scientific interest shown in the data being obtained; we had not prepared for the possibility of maintaining the data link, perhaps indefinitely. Also, there is pressure to extend the wave measuring network to other locations and to add wind measurements at appropriate sites. Our intention is to respond to these pressures by suggesting cooperation and by offering technological knowledge.

Finally, as we consider becoming "operational," we see as a continuing problem the decision to make the relay station active and the base station passive. That is, we have put the control of the link at the relay station: it looks at its own clock and decides when to start sending some data. This was a learning experience, as we had intended, but it is not the best way to be operational. If the control were firmly at the base station, including the command to start sampling and the command to start sending data, then the base could request the next data set whenever it wanted it, and could also get it more or less often if it prefered. This would allow the base to be shut down for maintainance, for example, without causing any lockup of the relay station's TNC-PL1000 combination. More importantly, the base could then be used for several tasks, or to obtain the data from several remote stations, without jeopardizing its being available at just that moment that the relay station decides to start sending data.

7.2 Scientific and Technical Extensions

Several useful extensions are alluded to in the section above:

<u>Science Extensions</u>

1. Maintain the wave data acquisition throughout the winter season, to give needed environmental information on the wave climate of Martha's Vineyard.
2. Add a telemetering wind sensor to the Buoy Farm mooring suite.

<u>Technical Extensions: Making the Present System Work Better</u>

3. Provide an accurate time signal to the Zenith computer, by use of the Heathkit "Most Accurate Clock" or by acquiring "internet time".
4. Fix or replace the Zenith computer to eliminate the floppy disc reading errors, or trap the error so the system can be self-correcting.
5. Fix or replace the Epson printer so the paper does not jam.

6. Change the protocol to have the relay station be passive; that is, put the control at the base station to trigger a sampling period by the PL1000, and to request the transmission of the data set.

7. Replace the 27 MHz analog transmission from the Waverider with a VHF packet radio transmission. This would require an A/D converter in the buoy, a TNC, and a new radio (a transceiver instead of just a transmitter). The relay station would become just a digital repeater station instead of a receiving and computing station. The base station would trigger a transmission of a data sequence of a specified length, for example 2048 points from half-second sampling of the wave displacement.

7.3 Conclusions

The two main goals of this project have been achieved. We have been able to automate the acquisition, processing, and dissemination of some telemetered oceanographic data, and we have gained some useful and informative experience with packet radio for ocean data telemetry.

The difficulties in the project have been mainly associated with interference on the 27 MHz analog transmissions from the Waverider buoy to the relay station on Martha's Vineyard, and with various minor and potentially correctable equipment problems.

The successes in the project have been in the ease of use of the packet radio and in its remarkable ability to pass error-free data, and in the computer networking to pass the processed data over Ethernet to a mainframe computer where the data can be accessed over public dial-up telephone lines.

Continued use of the system would enhance the scientific value of the wave data, since so far (April-September) the wave climate has been relatively benign. Because winter storms, especially in February, are the main cause of coastal erosion on Martha's Vineyard, we are considering leaving the system deployed through the coming winter.

A major technical extension of the project would be to replace the 27 MHz analog transmission in the Waverider buoy with a VHF packet radio transmission. This would eliminate data errors caused by interference. It could also increase the range of the transmissions because a gain antenna could be more easily used at both ends of the path, and more power could be put into the transmissions since there would be no need to transmit continuously.

8 References

Datwell ibv, "Operation and service Manual for Waverider, Supplemented for 600-9 series" Datawell bv, Laboratory for Instrumentation, Zomerluststraat 4, 2012 LM Haarlen, Netherlands.

Briscoe, M. G., "Status Report on Ocean Data Telemetry", Woods Hole Oceanographic Institution Technical Report WHOI-86-17, Woods Hole, MA, May 1986.

Briscoe, M. G., and D. E. Frye, "Motivations and Methods for Ocean Data Telemetry", *Marine Technology Society Journal*, Vol. 21, No.2 pp 42-57.

Frye, D. E., "Data Telemetry, Assimilation and Ocean Modeling-Semi Annual Report" Woods Hole Oceanographic Institution Technical Report WHOI 87-21, Woods Hole, MA, June 1987.

A. Gay Head Relay Van

A.1 PL1000 Digitizer Program

```
10 REM Program to digitize 2048 samples at 500 ms rate; each half-hour
11 REM      - M.Briscoe, 27 Apr 87
12 REM
19 REM flags for autostart;no echo;computer mode
20 SFL 1,1:SFL 16,1:SFL 21,1
25 PRINT "*** PL1000 Running"
29 N=2048
30 DBUF N+1
32 REM
33 REM check for start times at 51m28s or 21m28s past each hour
35 IF CLO(1)=51 THEN 37
36 IF CLO(1)=21 THEN 37 ELSE GOTO35
37 IF CLO(2)=28 THEN 40 ELSE GOTO35
40 R=CLO(5):S=CLO(4):T=CLO(0):U=CLO(1):V=CLO(3):V=V-1980
45 REM
46 REM get analog data; read voltage check
50 ASCAN 0,0,50 MS,N,B,0
51 ASCAN 1,1,1 MS,1,B,2048
52 REM
53 REM print headers,buffer,footers
54 D=CLO(5):H=CLO(0):M=CLO(1)
56 PRINT:PRINT "ZCZC  ";
57 IF C<100 THEN GOSUB 100
58 IF C<10 THEN GOSUB 100:PRINT C:C=C+1
59 IF C<=999 THEN GOTO63 ELSE C=0
63 PRINT "MMNT  ";"KMVY  ";
64 IF D<10 THEN GOSUB 100:PRINT D;
65 IF H<10 THEN GOSUB 100:PRINT H;
66 IF M=0 THEN PRINT "00"
67 IF M=0 THEN GOTO 69
68 IF M<10 THEN GOSUB 100:PRINT M
69 PRINT "ZZXX  ";
70 IF R<10 THEN GOSUB 100:PRINT R;
71 IF S<10 THEN GOSUB 100:PRINT S;V;SPC(2);
74 IF T<10 THEN GOSUB 100:PRINT T;
75 IF U=0 THEN PRINT "00/";
76 IF U=0 THEN GOTO 78
77 IF U<10 THEN GOSUB 100:PRINT U;"/";
78 PRINT "  74116  ";"07102"
79 PRINT "999"
83 FOR J=0 TO N-7 STEP 8
84 FOR I=1 TO 7:Z=BUF(J+I-1)
85 PRINT SPC(1);:GOSUB 200:PRINT Y;
86 NEXT I
88 NEXT J
90 PRINT "333  ";"44185  61616  8";
91 L=ABS(BUF(2048)):IF L<1000 THEN GOSUB 100 ELSE GOTO 94
92 IF L<100 THEN GOSUB 100 ELSE GOTO 94
93 IF L<10 THEN GOSUB 100 ELSE GOTO 94
94 PRINT L;"  69696=": PRINT "NNNN":PRINT
97 GOTO35
98 REM
99 REM special printing subroutines
100 PRINT "0";:RETURN
150 PRINT "-";:RETURN
160 PRINT SPC(1);:RETURN
198 REM
199 REM subroutine to block the data
200 Y=ABS(Z):IF Z>=0 THEN GOSUB 160 ELSE GOSUB 150
201 IF Y>=1000 THEN RETURN
202 GOSUB 100:IF Y>=100 THEN RETURN
203 GOSUB 100:IF Y>=10 THEN RETURN
204 GOSUB 100:RETURN
```

A.2 TNC Parameters (Relay and Base)

```
8BITCONV OFF
AX25L2V2 ON
AUTOLF   ON
AWLEN    7
AXDELAY  0
AXHANG   0
BEACON   EVERY 0
BKONDEL  ON
BTEXT    OFF
BUDLIST  OFF
LINK STATE IS: CONNECTED TO WHOI-1
CONPERM  OFF
CHECK    0
CLKADJ   0
CMDTIME  1
CMSG     ON
CPACTIME OFF
CR       ON
CTEXT    *** THIS IS THE REMOTE AT GAYHEAD
CANLINE  $18
COMMAND  $03
CALSET   157
CANPAC   $19
CONOK    ON
CONMODE  TRANS
CONSTAMP OFF
DAYUSA   OFF
DELETE   OFF
DWAIT    1
DIGIPEAT ON
ECHO     OFF
ESCAPE   OFF
FLOW     ON
FRACK    4
FULLDUP  OFF
HEADERLN OFF
HID      OFF
LCOK     OFF
LFADD    ON
LCALLS
LCSTREAM ON
MONITOR  OFF
MALL     OFF
MCON     OFF
MFILTER  $13
MRPT     OFF
MSTAMP   OFF
MYCALL   GAYHD
MYALIAS
MAXFRAME 4
MCOM     OFF
NEWMODE  ON
NOMODE   OFF
NUCR     OFF
NULF     OFF
NULLS    0
PACLEN   128
PARITY   3
PASS     $16
PASSALL  OFF
PACTIME  AFTER 10
RETRY    15
REDISPLA $12
RESPTIME 5
SCREENLN 80
SENDPAC  $0D
START    $11
STOP     $13
STREAMSW $7C
STREAMCA OFF
STREAMDB OFF
TRFLOW   OFF
TRIES    0
TRACE    OFF
TXDELAY  40
TXFLOW   ON
UNPROTO  OFF
USERS    1
XFLOW    ON
XMITOK   ON
XOFF     $13
XON      $11
```

```
8BITCONV OFF
AX25L2V2 ON
AUTOLF   OFF
AWLEN    7
AXDELAY  0
AXHANG   0
BEACON   EVERY 0
BKONDEL  ON
BTEXT    OFF
BUDLIST  OFF
Link state is: CONNECTED to GAYHD P
CONPERM  ON
CHECK    6
CLKADJ   0
CMDTIME_ 1
CMSG     OFF
CPACTIME OFF
CR       ON
CTEXT    THIS IS THE WHOI VHF PACKET SYSTEM
CANLINE  $18
COMMAND  $03
CALSET   157
CANPAC   $19
CONOK    OFF
CONMODE  CONVERSE
CONSTAMP OFF
DAYUSA   OFF
DELETE   OFF
DWAIT    1
DIGIPEAT OFF
ECHO     OFF
ESCAPE   OFF
FLOW     OFF
FRACK    4
FULLDUP  OFF
HEADERLN OFF
HID      OFF
LCOK     ON
LFADD    OFF
LCALLS
LCSTREAM ON
MONITOR  ON
MALL     ON
MCON     OFF
MFILTER  $13
MRPT     OFF
MSTAMP   OFF
MYCALL   WHOI-1
MYALIAS
MAXFRAME 2
MCOM     OFF
NEWMODE  ON
NOMODE   OFF
NUCR     OFF
NULF     OFF
NULLS    0
PACLEN   128
PARITY   3
PASS     $16
PASSALL  OFF
PACTIME  AFTER 10
RETRY    15
REDISPLA $12
RESPTIME 12
SCREENLN 80
SENDPAC  $0D
START    $11
STOP     $13
STREAMSW $7C
STREAMCA OFF
STREAMDB OFF
TRFLOW   OFF
TRIES    0
TRACE    OFF
TXDELAY  40
TXFLOW   OFF
UNPROTO  OFF
USERS    1
XFLOW    OFF
XMITOK   ON
XOFF     $13
XON      $11
```

A.3 PL1000 Clock Reset Instructions

There are times when the PL1000 clock and the computer clock have drifted significantly from "real" time, and need to be corrected. Also, occasionally to recover from a serious error, the PL1000 clock should be reset. The following describes the command sequence used to reset the PL1000 clock. The ! symbol indicates the beginning of a comment.

The PL1000 clock can be reset from the base using a terminal program to allow communication. Communications parameters in the terminal program should be set as follows: 1200 baud, 7 data bits, 1 stop bit, even parity. All communication with the PL1000 MUST BE done in capital letters, it does not recognize lower case letters. To reset the clock, the following sequence should be sent:

ˆ C <CR>	!	Repeat this sequence until the TNC "CMD": prompt is returned to the screen
ECHO ON	!	These packet commands allow easier viewing on the
AUTOLF ON		monitor.
CONV	!	sets to converse mode, from whence PL1000 is contacted.
ˆ Vˆ C <C>	!	Sends ˆ C through the TNC at WHOI to the Gayhd TNC and*PL1000*.
PRINT TIME	!	If time is returned, you know you've got the PL1000.
SCL 1,10	!	If time is wrong, the SCL (set clock) command is used to reset hours (0), minutes (1), or seconds (2). This example resets minutes to 10.
PRINT TIME	!	Prints newly corrected time. If still incorrect, use SCL to correct, otherwise, execute next command.
RUN	!	Restarts the Packet Link Controller program, will return * * * * * PL1000 Running * * ** on the monitor.
ˆ C <C>	!	Returns to WHOI TNC in command mode. Returns CMD: prompt.
AUTOLF OFF	!	Resets packet parameters to what they need to
ECHO OFF		be for automatic logging.

Now exit the terminal program and restart WRPROC at base, and data aquisition and logging will begin correctly.

B. Base Station

B.1 WRPROC Program Report

NAME:	WRPROC Copyright (C) 1987 by Woods Hole Oceanographic Institution. All rights reserved.
TYPE:	Main Program
PURPOSE:	To acquire data from a Waverider buoy at specified intervals, compute the power spectrum and some other statistics from the data, and send the results to a file-server.
MACHINE:	IBM PC or clone thereof
SOURCE LANGUAGE:	Microsoft C and assembler

DESCRIPTION: This program spends most of the time watching the clock. Every half hour (at about 8 minutes and 38 minutes past the hour), it receives a set of data, consisting mainly of 2048 wave heights. The program does some computations, outputs a summary to the printer, creates some files, sends copies of files to the Red VAX, and then waits for the next set of data. Program operation takes the following steps:

1. Do necessary initialization
2. Start loop, ended by user striking the F1 key
   a. Get data
   b. Check data, and convert to wave heights
   c. Do computations
   d. Create files on PC
   e. Send file to the VAX
   f. Printer output
3. End loop

Details of each step follow.

The initialization section includes the following:

1. Check to see if the printer is on-line and ready. If not, rings the terminal bell 10 times, displays a message on the screen, and stops.
2. Sends one header line to the printer.

3. Opens COM1 port.

4. Sends commands to the PL1000 to turn echo off and enter conversation mode, and then to start running.

5. Open a scratch file called TMPFIL.DAT.

The program then enters a dormant period. During this time, the program checks to see if one of the function keys has been pressed by the user. The function keys implemented in the current program version are:

F1 to stop program

F2 to save the raw data file on the Red VAX. The file name will be WRddtttt.DAT, where dd is the day of the month, and tttt is the record time, both in UTC.

F8 to display incoming data on the screen.

Except for F1, these options can be reversed by striking the corresponding function key again. The dormant period ends at 5 minutes or 35 minutes past the hour. At this time the program displays a suitable message on the screen, and enters a wait period.

At the start of the wait period, the program initializes the input interrupt routine, thus enabling input. It will wait up to 6 minutes for the first input record, and up to one minute for subsequent records. If no carriage return (signifying end of record) is received within these limits, the input is timed out, and the program prints a message and returns to the dormant state. As each record is received, the record is stored in the scratch file. When a record starting with 'NNNN' is received, input has been completed, and the interrupt routine is terminated.

The program then rewinds the scratch file, and reads and decodes the records. The format of the file is described in Figure 3. The items which are used (in addition to the actual data) are sequence number, transmit time, latitude, longitude, quadrant, buoyid, and voltage. If the transmit time is not the same as the time the first record was received, the PL1000 clock is reset. If the voltage is less than 1000, the terminal bell is rung 10 times, and a message is displayed on the screen.

The next step is to check the data for out-of-range values, and convert the input to meters. Any out-of-range (>5000) points are replaced with linearly interpolated values. However, it appears very unlikely that any such values will ever occur. The other computations are:

1. Find, and remove, the mean of the entire series.

2. Find first differences of the series:

$$X(I) = X(I+1) - X(I); \qquad X(N) = X(N-1)$$

- Compute the power spectrum, using 8 pieces of 256 points each. The spectral estimates are recolored by dividing each estimate by 4 sin (k/n) where k is the estimate number. Actually, the first four, and the last 64 points of the spectrum are not computed. This leaves 60 estimates, corresponding to frequencies from 4/128 to 63/128, or wave period from 32s to 2s.
- Some statistics are computed from the remaining spectral estimates

a.	m_0	=	$\sum_{n=4}^{63} E_n$	where E_n is the i'th spectral estimate.
b.	m_1	=	$\left(\sum_{n=4}^{63} n\ E_n\right)/T$	where T is the time length of one piece
c.	m_2	=	$\left(\sum_{n=4}^{63} n^2\ E_n\right)/T^2$	
d.	H_s	=	$[4]\sqrt{m_o};$	this is the significant wave height
e.	T_i	=	$m_0/m_1;$	this is the mean wave period
f.	T_z	=	$\sqrt{m_o/m_2};$	this is the zero-up-crossing wave period
g.	T_m	=	$1/f_m$	where f_m is the frequency of the maximum spectral estimate. T_m is called the peak period.

This completes the computations

Two files are maintained on a floppy disk on a daily basis. The file names are created from the date, and the files are opened in 'append' mode. This means that if the file already exists, it will be added to, and if it does not exist, it will be created. This technique results in a file for each day. One file contains only a summary record for each data set. The other file, in addition to a summary, contains the spectral estimates. The updated version of both the files is sent to the Red VAX, in PODA:[I33.REP1]. Complete file descriptions are included under OUTPUT, below.

The printer output includes values of computed parameters, and a line printer plot of the spectrum. A sample is included in Appendix B.3

INPUT:

The input from RS232 port 1 consists of 264 'records', each one terminated by a <CR> and <LF>. The first and last few records contain no wave data, and are ignored by the program. The <LF> characters are also ignored. A description of the record contents can be found in Figure 3.

The port configuration is:

1200 baud, 7 data bits, 1 stop bit, even parity

The input is done by an interrupt routine. Because of the shortness of some of the input records and the desire to display the input records, the processing cannot quite keep up with the input. Fortunately, the input is not 'real time', but is buffered on the other end. So, after each <CR>, the program sends an XOFF character, to stop input until it is ready for the next record, when it sends on XON. This works very well, with no appreciable slowing of the processing.

As the input records are read, they are stored in a scratch file. Then, when the last record has been read, the scratch file is rewound, and the records are processed. This file can then be transferred to the Red VAX for further processing if desired.

OUTPUT:

A. Summary File

A summary file is created on floppy disk for each day. The name of the file is created from the year, month, and day of the month. For example, the file for May 27, 1987 would be S870527.DAT. The file is opened in 'append' mode, so a new file is created for each day. One record is written to this file for each data set. The items in this record are:

1. Year, month, and day
2. Time, 24-hour format, in UTC
3. Root mean square wave height
4. Significant wave height
5. Mean wave period
6. Zero-up-crossing wave period
7. Average value for entire data set
8. Maximum value for data set, with mean removed
9. Minimum value for data set, with mean removed
10. Frequency of spectral peak

A sample of this file is included in Appendix B.2.

After computations for each half hour data set have been completed, this complete file is copied, via Ethernet, to the Red VAX, PODA:[I33.REP1]. The file name remains the same. If the PC has a hard disk, this file is also stored on that. If not, it is stored on a floppy disk.

B. Spectrum file

The name of this file is created in the same way as the summary file, except the first character is C (for coefficients) instead of S (for summary). The file is created the same way, giving one file per day. Also, the file is transferred to the same directory on the Red VAX. The file is stored on hard disk or floppy in the PC exactly as is for the summary file. The contents of the file are different. For each data set, there will be 12 records, as follows:

Record 1:

1. Time of observation, 12-hour clock, in UTC
2. Mean value of data set

Record 2:

1. Value of m_0 , the zero'th moment
2. Value of m1 , the first moment
3. Value of m2 , the second moment
4. Significant wave height
5. Mean wave period
6. Zero-up-crossing wave period

Records 3 - 12

Contain spectral estimates. The first one corresponds to
a frequency of 4/128, and the last one corresponds to
a frequency of 63/126.
A sample of this file is included in Appendix B.2.

C. Raw Data File

In addition to the two files of computed data, a file of the original 2048 values of sea surface height can be transmitted to the VAX

D. Printer

Printed output contains values of computed parameters, and a printer plot of the spectral estimates. A sample is included in Appendix B.3 At strategic times the program checks the printer status. If at one of these times, the printer is not available, the program displays a message on the screen, and sends all printer output to the screen. This is done because otherwise the program would get hung up waiting

for the printer. If the printer is not available when the program is first started, the terminal bell is sounded, a message is displayed, and the program stops. It is possible to redirect the printer output, so the program can be run if no printer is available. See USAGE, below.

USAGE:

These instructions apply to a IBM-PC clone with two 360k floppy disk drives One disk must contain the following files: wrproc.exe, and ftp.exe (from FTP Software).

This is the program disk, and will also be used for the scratch file. The second disk must be empty; this is the data disk.

After the system has been booted up, remove the DOS disk and put the program disk in Drive A, and the data disk in Drive B.

Start the program with the command: wrproc

optionally followed by one command line argument, to redirect the printer output. The argument, if present, must be the file or device name to which the printer output is to be sent. For example, to send printer output to the screen enter wrproc CON. To discard printer output, enter wrproc NUL. The program greets the user with the message:

> Program wrproc, waverider processing. Copyright (C) 1987 Woods Hole Oceanographic Institution. All rights reserved.

> Place an empty formatted disk in Drive B, and press <RETURN> to continue:

Do as requested. The program should proceed with no more attention.

Although this program is intended to run unattended, the IBM PC and MS-DOS were not really built for unattended operation. Things must be attended to at least once a week, and preferably every day when possible. Items which need attention are:

1. Printer
   a. Check for paper jams
   b. Be sure there is enough paper
2. Check for low voltage at the Gay Head site. If low, contact someone to fix the problem.

3. Check for reasonable mean values. Should be between .32 and .4. If out of this range, look for other problems. Could be caused by a power outage at Gay Head or by data transmission interference.

4. The clock on the PC has a tendency to drift. Once or twice a week it may need setting.

5. Once a week replace the floppy disk in Drive B with a new formatted disk. Copy the summary and spectrum files for the current day to the new disk.

SUBPROGRAMS USED:

The following table lists functions included in the program.

	Name	File	Return Value	Description
	wrproc	wrproc.c	none	Main program
	chrdy	timup.c	status	Checks if input record complete
	comput	comput.c	none	Finds spectral moments and other parameters
	convrt	convrt.c	no. interp.	Converts to wave heights, replaces bad values
	datdec	datdec.c	no. points	Decodes records
	datget	datget.c	no. points	Reads input and storec in scratch file
	prnt	doprnt.c	none	Prints computed parameters
	dumpf	dumpf.c	none	Dumps scratch file to printer
	fdmean	comput.c	mean(real)	Finds maximum, minimum, and mean
	fndscl	lpplot.c	max. scale	Determines scale for spectrum plot
*	funcno	funcno.asm	func. no.	Gets function key number
	getset	getset.c	none	Does program initialization
	interp	convrt.c	no. interp.	Interpolates one bad section
	intrp	convrt.c	no. interp.	Interpolates bad values in whole array
*	intrw	intrnu.asm	none	Input interrupt routine
	lpplot	lpplot.c	none	Spectrum plot on printer
	ouprt	ouprt.c	none	Writes string to port
	powspec	comput.c	status	Finds power spectrum
*	prnrdy	prtset.asm	status	Checks if printer ready
*	prtset	prtset.asm	none	Opens RS232 port
	putsml	putdat.c	none	Stores record in summary file
	putspc	putdat.c	none	Stores records in spectrum file
@	rfft05	rfft05.c	none	Does fft
	rmean	comput.c	none	Removes mean from one piece
	settim	settim.c	none	Sets time on PL1000
	timup	timup.c	status	Checks if time to start looking for data
	valert	doprnt.c	none	Alerts for low voltage
*	wprt	prtset.asm	none	Writes one character to port.

* Written in assembly language
@ Acquired from Eddie Scheer

The program spawns commands to run the program FTP.EXE, used for file transfer. This program is proprietary and cannot be copied to other computers.

ERROR CONDITIONS:

The following error conditions are checked by the program, and the corresponding action taken.

1. *The mean of a data set is less than .3 or greater than .42.* The program outputs an appropriate message to the screen, and skips the offending data set. No records are written to either the summary file or the spectrum file, and the raw data file is not transferred.

2. *The printer is off-line or out of paper.* If this occurs at the start of a run, the terminal bell is sounded ten times, a message is displayed on the screen, and the program stops. Otherwise, a message is displayed on the screen, and further printer output is redirected to the screen. The program must be stopped and restarted to direct output back to the printer.

3. *Low voltage at Gay Head is detected in the input data.* The terminal bell is sounded ten times, and a message is displayed on the screen. This situation requires attention.

4. *An input record is not received after a specified length of time.* The program outputs a message to the printer and the screen, and returns to the dormant state.

5. *One of the input records (except the first) has the incorrect format.* A message is output to the printer, and the contents of the input file are dumped to the printer.

PROGRAMMER: Mary M. Hunt

ORIGINATOR: Dick Payne

DATE: May, 1987

B.2 Sample Summary and Spectral Coefficient Files

EXAMPLE SUMMARY FILE

871207	0000	0.179	1.694	4.270	3.947	0.338	1.808	-1.589	0.219
871207	0030	0.143	1.515	4.301	4.025	0.335	1.806	-1.499	0.156
871207	0100	0.162	1.611	4.303	3.974	0.329	1.868	-1.670	0.219
871207	0130	0.143	1.514	4.342	3.987	0.328	1.482	-1.656	0.219
871207	0200	0.109	1.320	4.323	4.090	0.332	1.267	-1.175	0.227
871207	0230	0.156	1.579	4.157	3.804	0.330	1.603	-1.845	0.227
871207	0300	0.149	1.542	4.186	3.826	0.324	1.960	-1.710	0.211
871207	0330	0.154	1.571	4.353	3.944	0.325	1.727	-1.535	0.172
871207	0400	0.151	1.554	4.109	3.815	0.323	1.637	-2.007	0.211
871207	0430	0.148	1.537	4.236	3.921	0.333	1.559	-1.865	0.219
871207	0500	0.142	1.506	4.199	3.891	0.337	1.538	-1.386	0.180
871207	0530	0.151	1.554	4.222	3.822	0.333	1.844	-1.584	0.211
871207	0600	0.129	1.438	4.283	3.933	0.339	1.449	-1.681	0.164
871207	0630	0.144	1.520	4.125	3.803	0.332	1.583	-1.643	0.180
871207	0700	0.129	1.437	4.257	3.934	0.328	1.727	-1.950	0.180
871207	0730	0.137	1.479	4.312	3.981	0.334	1.511	-1.337	0.180
871207	0800	0.134	1.462	4.261	3.921	0.331	1.350	-1.870	0.156
871207	0830	0.123	1.400	4.154	3.784	0.328	1.569	-1.201	0.141
871207	0900	0.140	1.498	4.279	3.906	0.322	1.349	-1.783	0.180
871207	0930	0.118	1.376	4.176	3.817	0.328	1.599	-1.745	0.188
871207	1000	0.139	1.493	4.146	3.828	0.329	1.248	-1.656	0.227
871207	1030	0.137	1.480	4.225	3.873	0.325	1.912	-1.594	0.242
871207	1100	0.161	1.607	4.286	3.989	0.323	1.334	-1.837	0.219
871207	1130	0.157	1.587	4.093	3.833	0.327	1.775	-1.455	0.219
871207	1200	0.174	1.669	4.277	3.994	0.329	1.824	-1.941	0.211
871207	1230	0.165	1.625	4.171	3.904	0.316	1.811	-1.503	0.211
871207	1300	0.161	1.606	4.132	3.820	0.325	1.917	-1.695	0.211
871207	1330	0.182	1.706	4.179	3.890	0.329	1.937	-2.297	0.219
871207	1400	0.152	1.557	4.069	3.807	0.329	1.470	-1.719	0.211
871207	1430	0.135	1.469	3.915	3.687	0.329	1.507	-1.597	0.219
871207	1500	0.116	1.363	3.852	3.617	0.331	1.326	-1.565	0.234
871207	1530	0.090	1.199	3.811	3.540	0.331	1.274	-1.662	0.250
871207	1600	0.076	1.106	3.695	3.444	0.335	1.008	-1.140	0.281
871207	1630	0.071	1.063	3.742	3.456	0.340	1.006	-0.948	0.297
871207	1700	0.061	0.985	3.713	3.449	0.333	1.327	-1.175	0.266
871207	1730	0.058	0.962	3.827	3.521	0.335	1.083	-1.193	0.258
871207	1800	0.045	0.845	3.666	3.421	0.340	0.971	-0.885	0.273
871207	1830	0.040	0.797	3.778	3.461	0.333	0.960	-1.160	0.266
871207	1900	0.037	0.769	3.914	3.506	0.318	0.918	-0.952	0.031
871207	1930	0.037	0.768	3.646	3.353	0.336	0.881	-1.326	0.352
871207	2000	0.029	0.681	3.638	3.332	0.336	0.882	-1.084	0.320
871207	2030	0.038	0.783	4.160	3.628	0.334	0.826	-0.899	0.031
871207	2100	0.031	0.708	3.819	3.422	0.334	0.742	-0.731	0.031
871207	2130	0.026	0.642	3.794	3.407	0.328	0.701	-0.916	0.219
871207	2200	0.028	0.666	4.060	3.562	0.331	0.726	-1.114	0.031
871207	2230	0.031	0.701	3.755	3.319	0.329	1.101	-0.828	0.039
871207	2300	0.036	0.762	3.934	3.465	0.319	0.963	-1.527	0.039
871207	2330	0.039	0.794	3.649	3.307	0.332	1.041	-0.864	0.367

EXAMPLE SPECTRAL COEFFICIENT FILE

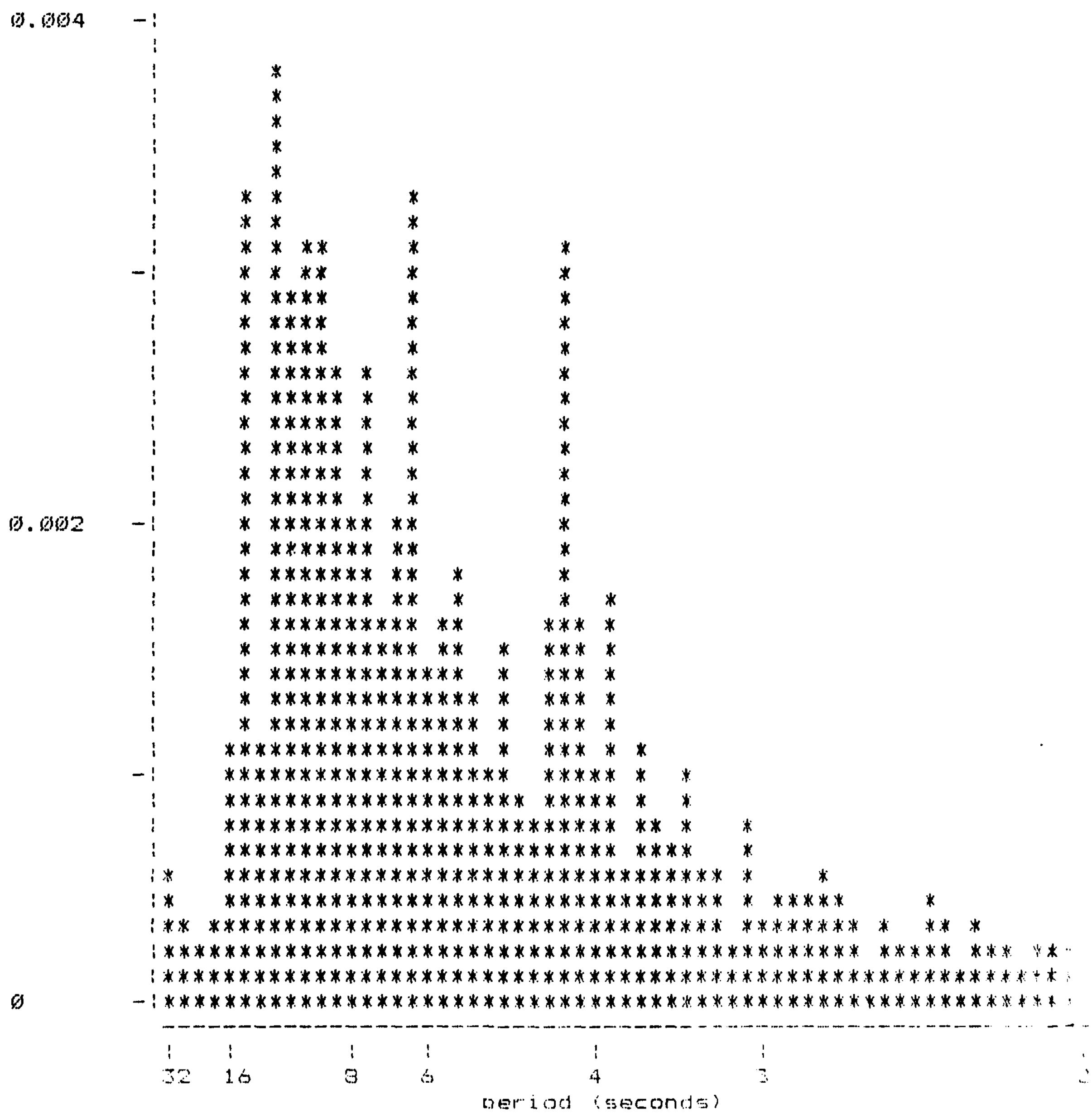
```
871207 0000 0.338
   1.794e-001  4.202e-002  1.151e-002  1.694e+000  4.270e+000  3.947e+000
   1.521e-003  2.123e-003  9.930e-004  4.544e-004  6.134e-004  5.251e-004
   1.266e-003  1.258e-003  6.518e-004  1.660e-003  1.714e-003  1.693e-003
   1.784e-003  2.406e-003  2.712e-003  3.712e-003  7.564e-003  7.172e-003
   8.451e-003  7.249e-003  7.699e-003  7.396e-003  6.070e-003  8.658e-003
   1.516e-002  4.873e-003  8.652e-003  3.287e-003  5.113e-003  3.192e-003
   3.670e-003  3.765e-003  2.419e-003  3.424e-003  2.931e-003  2.849e-003
   2.419e-003  1.702e-003  3.023e-003  1.064e-003  1.619e-003  1.097e-003
   1.111e-003  1.345e-003  1.571e-003  1.756e-003  2.688e-003  9.681e-004
   1.509e-003  1.064e-003  1.169e-003  9.299e-004  1.459e-003  9.414e-004
   1.387e-003  1.163e-003  1.814e-003  9.508e-004  9.573e-004  1.021e-003
871207 0030 0.335
   1.434e-001  3.335e-002  8.854e-003  1.515e+000  4.301e+000  4.025e+000
   4.081e-004  3.735e-004  5.811e-004  9.967e-004  2.212e-004  4.078e-004
   3.578e-004  6.921e-004  1.091e-003  1.049e-003  6.332e-004  1.625e-003
   1.186e-003  1.990e-003  1.693e-003  2.471e-003  9.172e-003  7.223e-003
   7.442e-003  5.578e-003  3.939e-003  7.944e-003  7.055e-003  6.176e-003
   6.658e-003  4.078e-003  7.264e-003  4.044e-003  4.604e-003  4.952e-003
   5.201e-003  2.250e-003  2.751e-003  3.003e-003  2.268e-003  1.458e-003
   1.095e-003  1.728e-003  1.628e-003  1.667e-003  1.633e-003  1.688e-003
   1.682e-003  8.227e-004  1.468e-003  9.174e-004  4.012e-004  1.086e-003
   9.147e-004  7.824e-004  9.322e-004  9.837e-004  7.792e-004  2.324e-004
   7.902e-004  2.304e-004  1.366e-003  6.920e-004  4.067e-004  6.495e-004
871207 0100 0.329
.
.
.
871207 2300 0.319
   3.630e-002  9.228e-003  3.024e-003  7.622e-001  3.934e+000  3.465e+000
   6.592e-004  1.363e-003  5.544e-004  4.989e-004  4.216e-004  5.727e-004
   5.408e-004  8.280e-004  6.770e-004  5.573e-004  6.833e-004  8.843e-004
   6.399e-004  4.463e-004  3.395e-004  1.142e-003  5.731e-004  1.058e-003
   6.078e-004  3.798e-004  6.253e-004  3.435e-004  6.668e-004  6.047e-004
   4.346e-004  5.735e-004  6.647e-004  4.998e-004  3.570e-004  8.270e-004
   2.668e-004  3.364e-004  6.474e-004  5.715e-004  3.443e-004  5.461e-004
   4.813e-004  4.069e-004  3.223e-004  7.230e-004  6.934e-004  8.518e-004
   4.351e-004  9.662e-004  9.320e-004  7.094e-004  8.565e-004  7.157e-004
   4.013e-004  9.158e-004  6.474e-004  4.990e-004  4.355e-004  7.243e-004
   8.591e-004  3.298e-004  4.078e-004  6.336e-004  3.272e-004  2.947e-004
871207 2330 0.312
   3.940e-002  1.080e-002  3.603e-003  7.940e-001  3.649e+000  3.307e+000
   1.147e-003  6.062e-004  7.178e-004  4.431e-004  3.117e-004  3.731e-004
   2.515e-004  3.437e-004  5.560e-004  7.961e-004  2.565e-004  5.421e-004
   3.233e-004  6.972e-004  7.398e-004  6.406e-004  5.856e-004  8.345e-004
   3.963e-004  5.108e-004  8.940e-004  6.360e-004  5.701e-004  4.726e-004
   4.722e-004  2.435e-004  6.152e-004  3.119e-004  4.496e-004  8.527e-004
   4.186e-004  3.132e-004  6.044e-004  3.686e-004  6.104e-004  7.428e-004
   1.002e-003  1.141e-003  6.628e-004  1.636e-003  1.475e-003  1.493e-003
   8.995e-004  2.014e-003  1.134e-003  8.393e-004  6.825e-004  5.866e-004
   6.167e-004  5.766e-004  8.175e-004  6.991e-004  3.341e-004  4.336e-004
   4.621e-004  5.622e-004  3.541e-004  2.843e-004  4.939e-004  5.546e-004
```

B.3 Sample Printer Output:

```
Waverider data received starting at GMT Fri Apr 15 16:38:20 1988
Spectral file transferred to Red VAX.
Summary file transferred to Red VAX.

Seq. no.    = 20      Significant wave height        =  1.019 meters
Buoyid      = 185     Mean wave period               =  5.39 seconds
Voltage     = 3251    Zero-up-crossing wave period   =  4.72 seconds
Quadrant    = 7       Peak period                    = 11.64 seconds
Latitude    = 4116    Maximum wave crest             =  0.873 meters
Longitude   = 7102    Minimum wave trough            = -0.915 meters
                      Mean value of series           =  0.325

Spectral plot (vertical scale in m*m/Hz):
```



B.4 Program

```
/*
        File  WRPROC.C
======================================================================
        Program wrproc        March, 1987

    This program collects data from a serial port, computes the FFT,
    and some statistics, stores things in various files, and sends
    a file to the VAX.
          Programmer:  Mary M. Hunt
          Originator:  Dick Payne
          Date:        March, 1987

   The functions called directly by wrproc are:
       getset           does needed initialization
       funcno()         returns value of function key struck, or zero if none
       timup(starter)   returns non-zero value when it is time to start
                        listening
       datget           gets to next set of data
       convrt           converts to wave heights and interpolates bad values
       powspec          computes power spectrum, etc.
       doprnt           prints computed parameters
       putspc           stores spectrum, etc. in disk file
       putsml           writes record to summary file, and sends to VAX
       lpplot           makes line printer plot
======================================================================*/

#include  <stdio.h>
#include  <math.h>
#include  <conio.h>
#include  <pspecs.inc>
#include  <chardef.inc>
#include  <strdef.inc>
#include  <process.h>

   main(argc, argv )
     int   argc;
     char  *argv[];
     {
     int  rtncd, nofunc, nbad, exstat,npts, good, savraw=0;
     int  rtncod, starter=0, prnuse=1;
     char comnd[35];
     FILE *prnout;
/*******************************************************************
        Set up optional command line argument
        to redirect printer output.
******************************************************************/

     if ( argc > 1 )
        {
         prnuse = 0;
         prnout = freopen ( argv[1], "w", stdprn );
         if ( prnout == NULL )
            printf ( "   No good.\n" );
        }
     shodat = 0;
     timrec = 0;
     getset();
/*******************************************************************
```

```
        Start main program loop.
*******************************************************************/

    while ( ( nofunc = funcno() ) != 1 )     /* Check for function key */
      {
/*******************************************************************
        Wait until time to start looking for data.
********************************************************************/
        if ( ( rtncod = timup(starter) )   )
          {
           printf ( "   Waiting for data at %s\n", ctime(&ltime) );
           starter = rtncod;
           good = 0;
           npts = datget(PORTNO, NOPTS);
           if ( npts == NOPTS )           /*  Check correct no. of points. */
             {
               nbad = convrt ( NOPTS );   /* Convert to wave heights   */
               if ( nbad < NOPTS/2 )      /* Check no. bad points      */
                 {
                   good = 1;
                   rtncd = powspec ( NOPTS );   /* Find spectrum       */
                   if ( rtncd == -1)
                     {
                       fprintf ( stdprn, "   Data set rejected, max = min.\r")
                       putc ( LINEFEED, stdprn );
                       good = 0;
                       doprnt ( nbad );
                     }
                   else if ( rtncd == -2 )
                     {
                       fprintf ( stdprn, "   Data set rejected. Mean value = %
                                 avrg );
                       good = 0;
                       doprnt ( nbad );
                     }
                   else
                     {
                        putspc ( nbad );             /*   Spectrum file   */
                        putsml ( nbad );             /*   Summary file    */
                        doprnt ( nbad );             /*   print parameters */
                        lpplot( nbad );              /*   Plot spectrum   */
                     }
                 }   /**********  End of loop checking  nbad  *************/
               else
                 {
                   good = 0;
                   fprintf ( stdprn, "   Too many bad values,  %d\r",nbad);
                   putc ( LINEFEED, stdprn );
                   fflush ( stdprn );
                 }

             }   /*****************  End of nopts = NOPTS loop   ********/
           rewind ( jnkf );
           exstat = 0;
          }    /********** end of  timup loop *******************/
/********************************************************************
           Check for F2 key - to send scratch file to VAX.
********************************************************************/
        if ( nofunc == 2 )
          {
```

```
            savraw = ~savraw;
            if ( savraw )
                printf ( "   Raw data transfer enabled.\n" );
              else
                printf ( "   Raw data transfer disabled.\n" );
           }

/**********************************************************************
           Send scratch file
**********************************************************************/
         if ( savraw && good && timrec )
            {
             sprintf ( comnd, "PUT TMPFIL.DAT DSKB:WR%021d%04d.DAT",
                       lmday, timrec );
             spawnlp ( P_WAIT, "ftp.exe", "ftp.exe", "-u", "E05_HCG1",
                       "REMSTAL","128.128.16.2", comnd, NULL );
             good = 0;
            }

/**********************************************************************
           Check for F8       Display raw data
**********************************************************************/
         if ( nofunc == 8 )
           {
            shodat = ~shodat;
            if ( shodat )
                printf ( "   Data display option turned on.\n" );
             else
                printf ( "   Data display option turned off.\n" );
           }
     }     /*********** End of while-loop  *****************/
    exit(exstat);
    }


/*
       File COMPUT.C
==========================================================================
     This file contains routines which compute the power spectrum, etc.
     They are:
           powspec     finds the power spectrum
           comput      finds the parameters
           rmean       removes mean from one piece
           fdmean      finds mean, max, min, of entire data set

     Programmer:  Mary Hunt
     Date:        May, 1987
=======================================================================*/

#include <stdio.h>
#include <math.h>
#include <pspecs.inc>

/* -----------------------------------------------------------------------
       Function to find power spectrum.
       Uses the following:
             fdmean
             rmean
             rfft05
```

```
                comput
-----------------------------------------------------------------------*/
powspec()
    {
      int     npiece, i, j, k, plen, pow;
      double  zed, pi = 3.1415926535;
      double  sin(double);
      float arg, xdiv, xd1, xd2;
      float fdmean();
/*
                        find first differences,
                        after removing mean.
                                                                 */
      avrg = fdmean();
      if ( wvmin == wvmax )        /* if max = min, spectrum will be all 0 */
         return(-1);               /*    so do not compute it.             */
      if ( avrg < MNMIN || avrg > MNMAX )
         return(-2);
      plen = PLEN;
      pow = POW;
      for ( i=0; i<NOPTS-1; i++ )
         datser[i] = datser[i+1] - datser[i];
      datser[NOPTS-1] = datser[NOPTS-2];
/*
                        set spectrum = 0
                                                                 */
      for ( i=0; i<NSPEST, spctrm[i] = 0; i++ );
/*
                        start piece loop
                                                                 */
      npiece = 0;
      for ( j = 0; j < NOPTS; j +=PLEN )
        {
         for ( i=0; i<plen; i++ )
            datuse[i] = datser[i+j];
         rmean ( datuse, plen );
         rfft05 ( datuse, &pow, &plen );
/*
                        add piece into spectrum
                                                                */
         for ( i=8,k=0; k<NSTOT; i+=2,k++ )
          {
           spctrm[k] = spctrm[k] + datuse[i]*datuse[i]
                        + datuse[i+1]*datuse[i+1];
          }
         npiece = npiece + 1;
        }
/*
                        end of piece loop.    Normalize.
                                                                */
      xd1 = 2.*npiece;
      xd2 = xd1 * plen;
      xdiv = xd2 * plen;
      for ( k=0; k<60; k++ )
       {
        arg = (k+4)*pi/plen;
        zed = sin(arg);
        spctrm[k] = spctrm[k]/(xdiv*zed*zed );
       }
      comput();
```

```
      return(0);
    }
/* *********************************************************************

           This function finds the desired parameters.

*********************************************************************/
   comput()
     {
       int    i, isav;
       float  sum, sum1, sum2;
       float  xmult, tlen;
       double  fndscl();

       tlen = PLEN * SAMINT;
       sum = 0.;
       sum1 = 0.;
       sum2 = 0.;

       for ( i=0; i<NSPEST; i++ )
         {
           xmult = i + 4;
           sum = sum + spctrm[i];
           sum1 = sum1 + xmult * spctrm[i];
           sum2 = sum2 + xmult*xmult*spctrm[i];
         }
       params[0] = sum;
       params[1] = sum1/tlen;
       params[2] = sum2/(tlen*tlen);
       params[3] = 4. * sqrt(params[0]);
       params[4] = params[0]/params[1];
       params[5] = sqrt ( params[0]/params[2] );
/*
              Find maximum spectral value, and plot scale
                                                          */
       isav = 0;
       spmax = spctrm[0];
       for ( i=1; i<NSPEST; i++ )
          {
           if ( spctrm[i] > spmax )
             {
               spmax = spctrm[i];
               isav = i;
             }
          }
       spscl = fndscl(spmax );
       hgfrq = ( isav+4) / 128.;
       hgper = 1.0/hgfrq;

       return;
    }
/* *********************************************************************

           Function to remove the mean from one piece.

*********************************************************************/
   rmean(dats, len )
     float  dats[];
     int     len;
     {
```

```
        double sum;
        int    i;
        float  avg;

        sum = 0.0;
        for ( i=0; i< len; i++ )
            sum = sum + dats[i];
        avg = sum/len;

        for ( i=0; i<len; i++ )
            dats[i] = dats[i] - avg;

        return;
    }

/* -------------------------------------------------------------------
        This function finds the maximum value, minimum value, and mean
        of the entire series.  It then subtracts the mean from the
        maximum and minimum.  The function value is the mean.
---------------------------------------------------------------------*/

  float fdmean ()
    {
      double  sum;
      int     i;
      float   avg;

      sum = 0.0;
      wvmax = 0.0;
      wvmin = 100000.;
      for ( i=0; i<NOPTS; i++ )
        {
         sum = sum + datser[i];
         if ( datser[i] > wvmax )
             wvmax = datser[i];
         if ( datser[i] < wvmin )
             wvmin = datser[i];
        }
      avg = sum/NOPTS;
      wvmax = wvmax - avg;
      wvmin = wvmin - avg;
      return(avg);
    }


/*
         File CONVRT.C
======================================================================
    This file contains the following:
            convrt
            intrp
            interp

          Programmer:  Mary M. Hunt
          Date:        April, 1987
======================================================================*/

#include  <stdio.h>
#include  <math.h>
```

```
#include  <conio.h>
#include  <pspecs.inc>

/*-----------------------------------------------------------------
      This function converts the input values to wave heights, and
      replaces any values > MAXVAL with MAXVAL.  It then calls intrp
      for interpolation.
-----------------------------------------------------------------*/

   convrt ( nopts )
     int   nopts;
     {
      int   nbad,i;
      for ( i=0; i<nopts; i++ )
        {
            if ( fabs(datser[i]) >= MAXVAL )
                    datser[i] = MAXVAL;
               else
                    datser[i] = datser[i]*CONFACT;
          }
       nbad = intrp();
       return(nbad);
     }

/*=================================================================
      Name:  interp
      Date:  March, 1987
      Programmer:  Mary Hunt
      Purpose:  To replace bad values in array 'datser' with
                interpolated values, and to count the number of
                interpolated values.  Bad values are those equal
                to MAXVAL.

      Bad values at the start are replaced by the first good value,
      and bad values at the end are replaced by the last good value.
      The routine uses linear interpolation for the rest.
=================================================================*/

#include <pspecs.inc>

  intrp()
    {
      int  i, istrt;
      int  nbad = 0;
      int  frstgd = 0;
      int  lastgd = NOPTS-1;
/*
             This part takes care of bad points at the start
                                                              */
      for ( i=0; i<NOPTS && datser[i++] == MAXVAL; )
         frstgd = i;
      for ( i=0; i<frstgd; i++ )
         {
          datser[i] = datser[frstgd];
          nbad = nbad + 1;
          }
/*
             This does the same for the end.
                                                              */
      for ( i=NOPTS-1; i>=0 && datser[i--] == MAXVAL; )
```

```
            lastgd = i;
        for ( i=NOPTS-1; i>lastgd; i-- )
            {
            datser[i] = datser[lastgd];
            nbad = nbad + 1;
            }
/*
              Now take care of the middle section.
                                                                     */
        for ( i=0; i<NOPTS; i++ )
            if ( datser[i] == MAXVAL )
              {   istrt = i-1;
                  nbad = nbad + interp(istrt);   }
        return(nbad);
      }

/*==========================================================================
                Function to do interpolation

     istrt  is the subscript of the last good value before at least
            one bad value.  This function finds how many bad values
            there are, and replaces them with linearly interpolated
            values.
========================================================================*/

   interp(istrt)
     int  istrt;
     {
        int   i, iend, j, nint;
        float  diff, perdiff;
        for ( i=istrt+1; i<NOPTS && datser[i] == MAXVAL;i++)
            ;
        iend = i;
        diff = datser[iend] - datser[istrt];
        perdiff = diff / ( iend - istrt );
        for ( j=1,i=istrt+1; i<iend; i++,j++ )
            datser[i] = datser[istrt] + perdiff*j;
        nint = iend - istrt - 1;
        return(nint);
     }


/*
        File DATDEC.C
==========================================================================
        This function reads records from the scratch file, one at a time,
        and decodes them.  If it can't understand a record, it returns
        a negative value.  Otherwise, it returns a value of zero.
        Other functions used are   valert, settim.

        Programmer:  Mary Hunt
        Date:        May, 1987
=========================================================================*/

#include <studio.h>
#include <math.h>
#include <pspecs.inc>
#include <chardef.inc>
```

```
datdec( nopts )
  {
    char  datbfr[52];
    int   nchrs, buflen, rtnval, mnstrt;
    int   ipntr, i, j, ip;
    long  xmitim, datim[4];

      buflen = 48;
      ipntr = 0;
/*
              First record has sequence #.
                                                                   */
      fgets ( datbfr, 50, jnkf );
      if ( memcmp(datbfr,scmd,4) == 0 )      /* if record starts with cmd:,*/
         fgets ( datbfr, 50, jnkf );          /*   skip the record          */
      if ( memcmp(datbfr,stchr,4) != 0 )
            seqno = -1;
        else
            sscanf ( &datbfr[6], "%3d", &seqno );
/*
              Next record has transmission time.
                                                                   */
      fgets ( datbfr, 50, jnkf );
      if ( memcmp(datbfr,strc2,4) != 0 )
          {
            fprintf ( stdprn, "  Unidentified record \r" );
            putc ( LINEFEED, stdprn);
            fflush ( stdprn);
            return(-1);
          }
      sscanf ( &datbfr[12], "%61d", &xmitim );
/*
              Next record has start time, lat, long.
                                                                   */
      fgets ( datbfr, 50, jnkf );
      if ( memcmp(datbfr,strc3,4) != 0 )
          {
            fprintf ( stdprn, "  Unidentified record \r" );
            putc ( LINEFEED, stdprn);
            fflush ( stdprn);
            return(-1);
          }
      for ( i=0, ip=6; i<4; i++, ip +=7 )
           sscanf ( &datbfr[ip], "%5ld", &datim[i] );
      latude = datim[2] % 10000;
      quadrnt = datim[2]/10000;
      longude = datim[3];
/*
              Record 4 has '999'.  Skip it.
                                                                   */
      fgets ( datbfr, 50, jnkf );
      if ( memcmp(datbfr,strc4,3) != 0 )
          {
            fprintf ( stdprn, "  Unidentified record \r" );
            putc ( LINEFEED, stdprn);
            fflush(stdprn);
            return(-1);
          }
/*
              Now we start the actual data.
```

```
                                                                    */
      fgets ( datbfr, 50, jnkf);
       while ( memcmp ( datbfr, three, 3) != 0)
        {
          for ( i=ipntr,j=0; j<buflen; i++,j+=6 )
             sscanf ( &datbfr[j], "%6f", &datser[i] );
          ipntr += 8;
          fgets ( datbfr, 50, jnkf);
        }
      rtnval = ipntr;
/*
              This record should have the buoyid and voltage.
                                                                    */
          for ( i=0, ip=5; i<4; i++, ip +=7 )
              sscanf ( &datbfr[ip], "%5ld", &datim[i] );
          buoyid = datim[0] % 1000;
          voltage = datim[2] % 10000;
          if ( voltage < 1000 )
              valert();
/*
              Start processing.
                                                                    */
      mnstrt = xmitim % 100;
      settim ( mnstrt );
      return(rtnval);
    }


/*
      File DATGET.C
=================================================================
      Function  datget:
      This function inputs and decodes one data set.
      It uses the following:
           intrw,enrupt     input interrupt routine
           chrdy            waits for next data record
           prnrdy           checks printer status
           datdec           decodes data set
           dumpf            dumps data set to printer

      The arguments are:
           int   port     RS232 port number (1 or 2)
           int   nopts    expected number of input data values

      Return value is the actual number of points, or < 0 for error.

      Programmer:  Mary Hunt
      Date:        May, 1987
====================================================================*/

#include <stdio.h>
#include <math.h>
#include <time.h>
#include <pspecs.inc>
#include <chardef.inc>

  datget( port, nopts )
    {
      char  inbfr[2][52];
```

```
int   nchrs, rtnst, buflen, secwt, npts;
int   isub, recnt;
long  ntime;

  buflen = 52;
  irdy = 0;
  wprt ( PORTNO, CQ );
  wprt ( PORTNO, CARET);
  wprt ( PORTNO, CARET);
  npts = 0;
/*
          Call to initialize the interrupt routine
                                                          */
  rewind ( jnkf );
  intrw ( PORTNO, inbfr, &irdy, &nchrs, &rtnst, buflen);
/*
          Get all data and~store in temp. file.
                                                          */
  secwt = 360;
  isub = 0;
  recnt = 0;
  strcpy ( inbfr, "JUNK" );
  while ( memcmp(&inbfr[isub][0], enchr, 2) != 0 )  /* this checks */
    {                                                 /* for NNNN    */
      if ( funcno() == 8 )     /* check for F8  */
           shodat = ~shodat;
      if ( ( isub = chrdy(secwt,&ntime ) ) < 0 )  /* wait until data */
           goto timout;                           /* ready or timeout*/
      irdy = 0;
      if ( nchrs > 1 && inbfr[isub][0] != QUEST )  /* skip record if */
         {                                      /* ? or no data       */
           secwt = 60;
           inbfr[isub][nchrs] = '\n';
           inbfr[isub][nchrs+1] = NULL;      /* add NULL and    */
           fputs ( &inbfr[isub][0], jnkf );   /* store in scratch file */
           recnt++
;
           if ( shodat )
               printf ( " %s", &inbfr[isub][0] );   /* display record */
           if ( recnt == 1 )
             ltime = ntime;   /* save start time of 1st record */
          }
    }        /*     found  'NNNN'      */
/*
          Turn off interrupts, and save start time.
                                                         */
  enrupt(PORTNO);
  mhtim = gmtime ( &ltime );
  timrec = ( mhtim -> tm_hour) * 100;
  if ( mhtim -> tm_min  > 30 )
      timrec = timrec + 30;
/*
        Check if printer available
                                                        */
  if ( prnrdy() != 0 )
      {
        printf ( "   Printer unavailable. Will output to screen.\n");
        freopen ( "CON", "w", stdprn );
      }
/*
```

```
        Output line to printer
                                                                              */
         fprintf ( stdprn, "   Waverider data received starting at GMT %s\r",
                  asctime(mhtim) );
         fflush ( stdprn );

         lmday = mhtim -> tm_mday;
         lmon = ( mhtim -> tm_mon ) + 1;       /*  create date      */
         lyear = mhtim -> tm_year;             /*  for file         */
         filtim = lyear*10000 + lmon*100 + lmday;  /*  name         */
         rewind ( jnkf);
         npts = datdec(nopts);            /*  decode file      */
         if ( npts != nopts )             /*  if an error,     */
            {
              dumpf ( recnt );            /*    dump file to printer  */
              npts = -1;
            }
         return(npts);
/*
                We go here for a timout.
                                                                        */ timout:
         fprintf ( stdprn, " Timed out after %d records\r", recnt);
         putc ( LINEFEED, stdprn );
         fflush ( stdprn);
         enrupt ( PORTNO);
         if ( recnt > 1 )
           dumpf ( recnt );
         return(-2);
     }


/*
          File DOPRNT.C
=========================================================================
          This file contains  doprnt  and  valert.
          Function doprnt outputs input and computed parameters
          to the printer.
          There is one argument:   int  nbad    number of interpolated
                                                 points

          Programmer:  Mary Hunt
          Date:        May, 1987
========================================================================*/

#include <stdio.h>
#include <pspecs.inc>
#include <chardef.inc>

   doprnt(nbad)
     int   nbad;
       {
          char   prbuf[81];
          int    i, k;
          putc( LINEFEED, stdprn );
          for ( i=0; i<80; i++ )
              prbuf[i] = BLNK;
          k = sprintf ( prbuf+3, "Seq. no.    = %d", seqno );
          prbuf[k+3] = BLNK;
          sprintf ( prbuf+30, "Significant wave height      = %6.3f meters\r",
```

```
                params[3]);
        fputs ( prbuf, stdprn );
        putc ( LINEFEED, stdprn );

        for ( i=0; i<80; i++ )
            prbuf[i] = BLNK;
        k = sprintf ( prbuf+3, "Buoyid     = %d", buoyid );
        prbuf[k+3] = BLNK;
        sprintf ( prbuf+30, "Mean wave period               = %5.2f seconds\r",
          params[4] );
        fputs ( prbuf, stdprn );
        putc ( LINEFEED, stdprn );

        for ( i=0; i<80; i++ )
            prbuf[i] = BLNK;
        k = sprintf ( &prbuf[3], "Voltage    = %d", voltage );
        prbuf[k+3] = BLNK;
        sprintf ( prbuf+30, "Zero-up-crossing wave period = %5.2f seconds\r",
                params[5] );
        fputs ( prbuf, stdprn );
        putc ( LINEFEED, stdprn );

        for ( i=0; i<80; i++ )
            prbuf[i] = BLNK;
        k = sprintf ( prbuf+3, "Quadrant   = %d", quadrnt );
        prbuf[k+3] = BLNK;
        sprintf ( prbuf+30, "Peak period                    = %5.2f seconds\r",
            hgper );
        fputs ( prbuf, stdprn );
        putc ( LINEFEED, stdprn );

        for ( i=0; i<80; i++ )
            prbuf[i] = BLNK;
        k = sprintf ( prbuf+3, "Latitude   = %d", latude );
        prbuf[k+3] = BLNK;
        sprintf ( prbuf+30, "Maximum wave crest             = %6.3f meters\r",
            wvmax );
        fputs ( prbuf, stdprn );
        putc ( LINEFEED, stdprn );

        for ( i=0; i<80; i++ )
            prbuf[i] = BLNK;
        k = sprintf ( prbuf+3, "Longitude  = %d", longude );
        prbuf[k+3] = BLNK;
        sprintf ( prbuf+30, "Minimum wave trough            = %6.3f meters\r",
             wvmin );
        fputs ( prbuf, stdprn );
        putc ( LINEFEED, stdprn );

        putc ( LINEFEED, stdprn );
        fprintf ( stdprn, "   Spectral plot (vertical scale in m*m/Hz):\r" );
        putc ( LINEFEED, stdprn );
        putc ( LINEFEED, stdprn );
        fflush ( stdprn );

        return;
    }

/*-------------------------------------------------------------------------------
   This routine is called when the voltage is < 1000.
```

```
    It rings the terminal bell 10 times and prints a warning message.
---------------------------------------------------------------------*/

  valert()
    {
     int    i;
     char   warn[3];

     warn[0] = BELL;  warn[1] = NULL;
     for ( i=0; i<10; i++ )
        printf ( "%s", warn );
     fprintf ( stdprn, "   ***  Warning:  low voltage  ***********\r" );
     return;
    }



/*
      File DUMPF.C
======================================================================
      Function  dumpf
      This function dumps contents of the scratch file to the printer.
      It is called when there is some kind of input error.

      One argument:      int  recnt     Number of records in file.

      Programmer:  Mary Hunt
      Date:        May, 1987
======================================================================*/

#include <stdio.h>
#include <chardef.inc>
#include <pspecs.inc>

  dumpf(recnt)
    int   recnt;
    {
     int  i;
     long lctm;
     char datbfr[52];

     rewind(jnkf);
     time ( &lctm);
     putc ( LINEFEED, stdprn);
     fprintf ( stdprn, "File dump at  %s\r", ctime(&lctm) );
     putc ( LINEFEED, stdprn);
     putc ( LINEFEED, stdprn);
     fflush ( stdprn);

     for ( i=0; i<recnt; i++ )
        {
         fgets ( datbfr, 50, jnkf);
         fprintf ( stdprn, "%s\r", datbfr );
         putc ( LINEFEED, stdprn);
        }
     putc ( FFEED, stdprn );
     fflush ( stdprn );
     return;
    }
```

```
/*
        File GETSET.C
======================================================================
        Function getset()
   This function does preliminary initialization tasks:
      1.  Checks if printer ready.
      2.  Outputs message  to terminal and printer
      3.  Opens RS232 port
      4.  Sends initial characters to get port ready.
      5.  Opens scratch file.

   Uses the following routines:
      prnrdy
      prtset
      wprt

   Programmer:  Mary Hunt
   Date:        May, 1987
======================================================================*/

#include <time.h>
#include <stdio.h>
#include <pspecs.inc>
#include <chardef.inc>

  getset(prnuse )
    int  prnuse;
    {
      long   strtim, nextim;
      char   oubf[8], warn[3];
      int    i;

      time(&strtim);
      mhtim = gmtime(&strtim);
/********************************************************************
                 Check if printer is ready.
  ********************************************************************/
      if ( prnrdy() != 0 && prnuse )
          {
            warn[0] = BELL;  warn[1] = NULL;
            for ( i=0; i<10; i++ )   printf ( "%s", warn );
            printf ( "   Printer not ready.  Please fix and restart program.\n")
            exit(1);
          }
/* *******************************************************************
                 Display greeting to user
   *******************************************************************/

      printf ( "   Program wrproc, waverider processing.\n" );
      printf ( "   Copyright (C) 1987, Woods Hole Oceanographic Institution.\n")
      printf ( "   All rights reserved.\n\n" );
      printf ( "   Place an empty formatted disk in Drive B,\n");
      printf ( "   and press <RETURN> to start.\n");
      getchar();

/* *******************************************************************
                 Send one line to printer
```

```
*************************************************************************/
      fprintf (stdprn, " Program wrproc, start of run %s\r",
               asctime(mhtim) );
      putc ( LINEFEED, stdprn );
      putc ( LINEFEED, stdprn );
      fflush ( stdprn);

      prtset ( BAUD, PORTNO );          /* *** Sets up COM1     **** */

      wprt ( PORTNO, CQ );
      wprt ( PORTNO, CARET );
/*********************************************************************
              This next code sets echo off and conversation mode
******************************************************************** */
      wprt ( PORTNO, CC );         /*    Puts PL1000         */
      wprt ( PORTNO, CC );         /*      in command        */
      wprt ( PORTNO, CC );         /*         mode.          */

/* *******************************************************************
             Wait at least one second
   *******************************************************************/
      time(&strtim);
      do
         time(&nextim);
      while ( nextim-strtim < 2 );

      sprintf ( oubf, "E OFF" );              /*    Turn          */
      oubf[5] = CARET;  oubf[6] = NULL;       /*      ECHO        */
      ouprt ( PORTNO, oubf );                 /*        OFF       */

      sprintf ( oubf, "CONV" );               /*    Put PL1000    */
      oubf[4] = CARET;  oubf[5] = NULL;       /*      in conversation    */
      ouprt ( PORTNO, oubf );                 /*          mode    */
/*********************************************************************
               Tell PL1000 to go
*********************************************************************/
      wprt ( PORTNO, CV );
      wprt ( PORTNO, CQ );
/* *********************************************************************
               Open scratch file    TMPFIL.DAT
********************************************************************/

      jnkf = fopen ( tfname, "w+");
      if ( jnkf == NULL )
         {
           printf ( "   Cannot open scratch file.\n" );
           exit(1);
         }
      return;
    }


/*
         File LPPLOT.C
=========================================================================
         This file contains:
            lpplot     line-printer plot of spectrum
            fndscl     determines plot scale
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
```

```
        Function lpplot, creates printer plot of spectrum.
        Checks for printer out of paper (happens in middle of plot.).

        Programmer:  Mary Hunt
        Date:        May, 1987
==========================================================================*/

#include <stdio.h>
#include <math.h>
#include <pspecs.inc>
#include <chardef.inc>

  lpplot( nbad )
    int   nbad;
    {
      int    i, j, dotinx, isav,k;
      float  dotpos;
      int    nlin = 40;
      char   prbuf[81];
      double fndscl();
      unsigned  pchk, prnrdy();

/*
              Fill buffer with blanks.
                                                                    */
    for ( i=0; i<NSPEST; i++ )
        {
         for ( j=0; j<nlin; j++ )
            spot[j][i] = ' ';
        }
/*
             Scale spectrum and store in print buffer.
                                                                   */
      for ( i=0; i<NSPEST; i++ )
        {
          dotinx = nlin*spctrm[i]/spscl;
          if ( dotinx == nlin )
              dotinx = nlin-1;
          for ( j = dotinx; j>= 0; j-- )
              spot[j][i] = '*';
        }
/*
             Now, we encode one row at a time, and output to
             the printer.  In addition to the actual spectrum, there
             are tic marks and labels for the vertical axis, and
             the actual axis.
                                                    */
      putc ( CARET, stdprn);
      putc ( LINEFEED, stdprn );
      for ( j=nlin-1; j>=0; j-- )
        {
          for ( i=0; i<81; i++ )       /*  blank the print line  */
              prbuf[i] = BLNK;
          if ( j == nlin-1 )           /*  first line            */
            {
              k = sprintf ( prbuf, "%lg", spscl );
              prbuf[k] = BLNK;
              prbuf[8] = MINUS;
            }
          if ( j == (nlin-1)/2 )       /*   middle line    */
```

```
            {
             k = sprintf ( prbuf, "%lg", spscl/2.0 );
             prbuf[k] = BLNK;
             prbuf[8] = MINUS;
            }
        if ( j == (nlin-1)/4 || j == 3*(nlin-1)/4 )  /* 1/4 or 3/4  */
             prbuf[8] = MINUS;
        if ( j == 0 )                    /*  last line of spectrum      */
            {
             k = sprintf ( prbuf, "%lg", 0.0 );
             prbuf[k] = BLNK;
             prbuf[8] = MINUS;
            }
        prbuf[9] = ULINE;

        for ( i = 0; i<NSPEST+1; i++ )
             prbuf[i+10] = spot[j][i];
        prbuf[72] = NULL;
/*
                   Chack printer
                                                                 */
        pchk = prnrdy();
        if ( pchk != 0 && pchk != 0X8000 )
             {
              printf ( "   Printer unavailable. Will output to screen.\n");
              freopen ( "CON", "w", stdprn );
             }

        fputs ( prbuf, stdprn );      /*  This outputs the line    */
        putc ( CARET, stdprn );
        putc ( LINEFEED, stdprn );
       }
/*
                 This gives a line of under-score characters
                                                                 */
    for ( i=0; i<10; i++ )
       prbuf[i] = BLNK;
    for ( i=10; i<NSPEST+11; i++ )
       prbuf[i] = USCR;
    prbuf[72] = NULL;
    fputs ( prbuf, stdprn );
    putc ( CARET, stdprn );
    putc ( LINEFEED, stdprn );
/*
                This makes tick marks on period axis
                                                                 */
    for ( i=0; i<80; i++ )
        prbuf[i] = BLNK;
    prbuf[10] = ULINE;
    prbuf[14] = ULINE;
    prbuf[22] = ULINE;
    prbuf[27] = ULINE;
    prbuf[38] = ULINE;
    prbuf[49] = ULINE;
    prbuf[70] = ULINE;
    prbuf[72] = NULL;
    fputs ( prbuf, stdprn );
    putc ( CARET, stdprn );
    putc ( LINEFEED, stdprn );
/*
```

```
                Label the tick marks
                                                                    */
        for ( i=0; i<80; i++ )
            prbuf[i] = BLNK;
        k = sprintf ( &prbuf[10], "32" );
        prbuf[10+k] = BLNK;

        k = sprintf ( &prbuf[14], "16" );
        prbuf[k+14] = BLNK;

        k = sprintf ( &prbuf[22], "8" );
        prbuf[k+22] = BLNK;

        k = sprintf ( &prbuf[27], "6" );
        prbuf[k+27] = BLNK;

        k = sprintf ( &prbuf[38], "4" );
        prbuf[k+38] = BLNK;

        k = sprintf ( &prbuf[49], "3" );
        prbuf[k+49] = BLNK;

        sprintf ( &prbuf[70], "2" );
        fputs ( prbuf, stdprn );
        putc ( CARET, stdprn );
        putc ( LINEFEED, stdprn );
/*
                  Axis label
                                                                    */
        for ( i=0; i<80; i++ )
            prbuf[i] = BLNK;
        sprintf ( &prbuf[31], "period (seconds)" );
        fputs ( prbuf, stdprn );
        putc ( CARET, stdprn );
        putc ( FFEED, stdprn );
        putc ( LINEFEED, stdprn );

        fflush ( stdprn);

        return;
      }

    double fndscl ( maxsp )
      float  maxsp;
      {
        float  dec, xmd;
        double  ymax;
        int     intop, mextop;
        double dsmax, dextop, ten = 10.;

        dsmax = maxsp;
        dec = log10 ( dsmax );
        if ( dec < 0. )
          dec = dec - 1.0;
        mextop = dec;
        dextop = mextop;
        xmd = pow ( ten, dextop );
        intop = maxsp / xmd + 1.0;
        ymax = intop * xmd;
```

```
    return ( ymax );
  }



/*
         File OUPRT.C
=====================================================================
         Function  ouprt
         Outputs a string to an RS232 port.
         Arguments are:
            int   port     Rs232 port number (1 or 2)
            char  oubf[]   String to be output to port
                            (must terminate with NULL)
         Uses function wprt to output each character.

         Programmer:  Mary Hunt
         Date:        May, 1987
======================================================================*/

#include <stdio.h>

  ouprt ( port, oubf )
    int  port;
    char oubf[];
    {
      int  i;

      i = 0;
      while ( oubf[i] != NULL )
        {
          wprt ( port, oubf[i] );
          i++;
        }
      return;
    }



/*
          File PUTDAT.C
==================================================================
          Routines to create the required files and store the
          data in them.

          Programmer:  Mary Hunt
          Date:        March, 1987
==================================================================*/

#include <time.h>
#include <stdio.h>
#include <pspecs.inc>
#include <process.h>
#include <chardef.inc>

/*========================================================================
   This function stores spectral estimates in a file.  The file name is
   created from C, followed by 2 digits each of year, month, and day.
   File type is  .DAT.  File is opened 'append', so all spectra for a
```

```
   day are in the same file.
========================================================================*/

   putspc(nbad)
    int nbad;
     {
      FILE   *fp;
      int    i, j, k;
      char   comnd[35];

      sprintf ( fname, "C%06ld.DAT", filtim );
      sprintf ( fnamb, "B:C%06ld.DAT", filtim );
      if ( ( fp = fopen(fnamb, "a" ) ) != NULL )
          {
            fprintf ( fp, "%04d %5.3f\n", timrec, avrg );
            for ( i=0; i<NPARM; i++ )
                fprintf ( fp, "%12.3e", params[i] );
            fprintf ( fp, "\n" );

            for ( i=0; i<NSPEST; i +=6 )
              {
                for ( j=i, k=0; k<6; j++,k++ )
                   fprintf ( fp, "%12.3e", spctrm[j] );
                fprintf ( fp, "\n" );
              }
            fclose ( fp );
/*
                Send to Red VAX
                                                                        */
            sprintf ( comnd, "PUT %s %s", fnamb, fname );
            k = spawnlp ( P_WAIT, "ftp.exe", "ftp.exe", "-u",
                          "I33_REP1", "DICK", "128.128.16.2",
                          comnd, NULL );
            if ( k == 0 )
               {
                printf ( "   Transferred spectral file.\n" );
                fprintf ( stdprn, "   Spectral file transferred to Red VAX.\r");
               }
              else
               {
                 printf ( "   Spectrum transfer failed.\n" );
                 fprintf ( stdprn, "   Spectral file transfer failed.\r");
               }
             putc ( LINEFEED, stdprn );
             fflush ( stdprn );
          }
/* ************************************************************************
                 Could not open file
***********************************************************************/

       else
         {
          printf ( "   Unable to open spectrum file.\n" );
          fprintf ( stdprn, "   Unable to open spectrum file.\r");
          putc ( LINEFEED, stdprn );
         }
      return;
    }
/* ========================================================================
     This function opens summary file, stores one record in it,
```

```
    and sends updated file to VAX.  File name, etc. same as for
    spectrum file, except starts with  S  instead of  C.

  ==================================================================*/
 putsml(nbad)
   int   nbad;
    {
     FILE   *fp;
     int    i, k;
     char   comnd[35];

     sprintf ( fnam2, "S%061d.DAT", filtim );
     sprintf ( fnamb2, "B:S%061d.DAT", filtim );
     if ( ( fp = fopen ( fnamb2, "a" ) ) != NULL )
        {
         fprintf ( fp, "  %061d", filtim );
         fprintf ( fp, "  %04d %6.3f", timrec, params[0] );
         for ( i=3; i<6; i++ )
            fprintf ( fp, "%8.3f", params[i] );
         fprintf ( fp, "%8.3f", avrg );
         fprintf ( fp, "%8.3f", wvmax );
         fprintf ( fp, "%8.3f", wvmin );
         fprintf ( fp, "%8.3f\n", hgfrq );
         fclose(fp);
/*
           Send summary file to file-server (Red VAX for now.)
                                                                */
         sprintf ( comnd, "PUT %s %s", fnamb2, fnam2 );
         k = spawnlp ( P_WAIT, "ftp.exe", "ftp.exe", "-u",
                    "I33_REP1", "DICK", "128.128.16.2",
                    comnd, NULL );
         if ( k == 0 )
           {
            printf ("   Transfer successful\n" );
            fprintf (stdprn,"   Summary file transferred to Red VAX.\r");
           }
         else
           {
            printf ( "   Transfer failed\n" );
            fprintf (stdprn,"   Summary file transfer failed.\r");
           }
         putc ( LINEFEED, stdprn );
         fflush ( stdprn );
        }
/***********************************************************************
           Cannot open summary file
*********************************************************************/
       else
        {
         printf ( "   Unable to open summary file.\n");
         fprintf ( stdprn, "   Unable to open summary file.\r");
         putc ( LINEFEED, stdprn );
        }
       return;
     }


/*
       File  RFFT05.C
```

```
------------------------------------------------------------------------
      This is a function to do a FFT on a time series of real numbers.
      All arguments are passed as pointers.  The arguments are:
            x     -  array of real numbers
            pow   -  power of 2
            len   -  length of series
      len must = 2**pow.

      Coefficients are returned in the order  cos, sine, cos, sine, etc.

      Acquired from Eddie Scheer, April 1987
------------------------------------------------------------------------*/

#include <math.h>
#include <stdio.h>

rfft05 (x,pow,len)
float *x;
long *pow,*len;
{
        double wr,wi,arg;
        int length,power,length2;

        length= *len;
        power= *pow;
        length2 = length;
        length/=2;
        power--;

        /* bit reverse */
        {
                register int i,mask,j;
                register float *p1,*p2,t;

                for (i=2; i<length2-2; i+=2) {
                        for (mask=2, j=0; mask<length2; mask<<=1) {
                                if (i & mask) j++;
                                j <<= 1;
                        }
                        if (i<j) {
                                p1 = x+i;
                                p2 = x+j;
                                t = *p1;
                                *(p1++) = *p2;
                                *(p2++) = t;
                                t = *p1;
                                *p1 = *p2;
                                *p2 = t;
                        }
                }
        }
        {
                register double tr,ti,ur,ui;
                register int le,i;
                register float *p1r,*p1i,*p2r,*p2i;
                int j,l,le2;

                le=2;
                for (l=0; l<power; l++) {
                        le <<= 1;
```

```
                le2 = le>>1;
                ur=1.0;
                ui=0.0;
                arg = 3.141592653589793 / (le2>>1);
                wr=cos(arg);
                wi= -sin(arg);
                for (j=0; j<le2; j+=2) {
                        p1r = x+j;
                        p1i = p1r+1;
                        p2r = p1r+le2;
                        p2i = p2r+1;
                        for (i=j; i<length2; i+=le) {
                                tr = *p2r * ur - *p2i * ui;
                                ti = *p2r * ui + *p2i * ur;
                                *p2r = *p1r - tr;
                                *p2i = *p1i - ti;
                                *p1r += tr;
                                *p1i += ti;
                                p1r += le;
                                p1i += le;
                                p2r += le;
                                p2i += le;
                        }
                        tr = ur*wr - ui*wi;
                        ui = ur*wi + ui*wr;
                        ur = tr;
                }
        }
}
{
        register float *p1r,*p1i,*p2r,*p2i;
        register double ur,ui,t,xer,xei,xor,xoi;
        register int k;
        ur=1.0;
        ui=0.0;
        arg = 3.141592653589793 / length;
        wr=cos(arg);
        wi= -sin(arg);
        x[length2]=x[0];
        x[length2+1]=x[1];
        p1r = x;
        p1i = x+1;
        p2r = x+length2;
        p2i = p2r+1;
        for (k=0; k<=length; k+=2) {
                xer=(*p1r + *p2r)/2.0;
                xei=(*p1i - *p2i)/2.0;
                xor=(*p1i + *p2i)/2.0;
                xoi=(*p2r - *p1r)/2.0;
                t = xor*ur - xoi*ui;
                xoi = xor*ui + xoi*ur;
                xor = t;
                *(p1r++) = xer + xor;
                p1r++;
                *(p1i++) = xei + xoi;
                p1i++;
                *(p2r--) = xer - xor;
                p2r--;
                *(p2i--) = xoi - xei;
                p2i--;
```

```
                        t = ur*wr - ui*wi;
                        ui = ur*wi + ui*wr;
                        ur = t;
                }
        }
}


/*
File SETTIM.C
====================================================================
Function  settim
This function checks to see if the PS time and the PL1000 time
are the same (to the minute).
If not, it resets the PL1000 time.
One argument:    int  mnstrt      start minute of xmission

Uses function  ouprt   to send messages to PL1000.

Programmer:  Mary Hunt
Date:        May, 1987
======================================================================
*/

#include <time.h>
#include <stdio.h>
#include <pspecs.inc>
#include <chardef.inc>

  settim(mnstrt)
int   mnstrt;
{
struct  tm  *timadr;
int     min, hour, year, month, mday, secnd, wday, rtnval;
char    oubf[15];
long    ntime;

rtnval = 0;
timadr = gmtime(&ltime);
min = timadr -> tm_min;

      if ( min != mnstrt )
        {
          time(&ntime);
          timadr = gmtime(&ntime);
          min = timadr -> tm_min;
/*    I just set the minute, skip hour, day, etc.      */
/*        hour = timadr -> tm_hour;
          mday = timadr -> tm_mday;
          month = ( timadr -> tm_mon ) + 1;
          year = timadr -> tm_year;            */
          secnd = timadr -> tm_sec;
/*        wday = timadr -> tm_wday;
          printf ( "  wday = %d", wday );      */

          oubf[0] = CQ; oubf[1] = CARET; oubf[2] = CARET; oubf[3] = LINEFEED;
          oubf[4] = NULL;
          ouprt ( PORTNO, oubf );
```

```
        oubf[0] = CV; oubf[1] = CQ;
        ouprt ( PORTNO, oubf );

        oubf[0] = CQ; oubf[1] = CARET;
        ouprt ( PORTNO, oubf );

        sprintf ( oubf, " CONV" );
        oubf[0] = CQ; oubf[5] = CARET; oubf[6] = LINEFEED; oubf[7] = NULL;
        ouprt ( PORTNO, oubf );

        oubf[0] = CV; oubf[1] = CC; oubf[2] = CARET; oubf[3] = LINEFEED;
        oubf[4] = NULL;
        ouprt ( PORTNO, oubf );
/*     Again, skip year, month, etc.                              */

  /*    sprintf ( oubf, "SCL 3,19%d", year );
        oubf[10] = CV; oubf[11] = CARET;  oubf[12] = NULL;
        ouprt ( PORTNO, oubf );

        sprintf ( oubf, "SCL 4,%3d", month );
        oubf[9] = CV; oubf[10] = CARET; oubf[11] = NULL;
        ouprt ( PORTNO, oubf );

        sprintf ( oubf, "SCL 5,%3d", mday );
        oubf[9] = CV; oubf[10] = CARET; oubf[11] = NULL;
        ouprt ( PORTNO, oubf );

        sprintf ( oubf, "SCL 6,%2d". wday );
        oubf[8] = CV; oubf[9] = CARET; oubf[10] = NULL;
        ouprt ( PORTNO, oubf );

        sprintf ( oubf, "SCL 0,%3d", hour );
        oubf[9] = CV; oubf[10] = CARET; oubf[11] = NULL;
        ouprt ( PORTNO, oubf );                              */

        sprintf ( oubf, "SCL 1,%3d", min );      /* This sets minute  */
        oubf[9] = CV; oubf[10] = CARET; oubf[11] = NULL;
        ouprt ( PORTNO, oubf );

        sprintf ( oubf, "SCL 2,%3d", secnd );   /*  and second  */
        oubf[9] = CV; oubf[10] = CARET; oubf[11] = NULL;
        ouprt ( PORTNO, oubf );

        sprintf ( oubf, "RUN" );
        oubf[3] = CARET; oubf[4] = LINEFEED; oubf[5] = NULL;
        ouprt ( PORTNO, oubf );

        fprintf ( stdprn, "   Time reset to %s\n", ctime(&ntime ) );
        putc ( CARET, stdprn);
/*      putc ( LINEFEED, stdprn );                    */
        fflush ( stdprn );
        printf ( "  Time reset to %s\n", ctime(&ntime) );
     }
   return ( rtnval );
  }


/*
        File TIMUP.C,UP
```

```
=====================================================================
         Function  timup
  This function tells the program when to start looking for data.
  If 'starter' = 0 (first time), it waits until either 5 minutes
  or 35 minutes past the hour.  If 'starter' = 1, it waits until
  5 minutes past the hour, and if 'starter' = 2, it waits until
  35 minutes past the hour.  In any case, the return value should
  be the next value for 'starter'.

         Programmer:  Mary Hunt
         Date:        April 1, 1987
=====================================================================*/

#include <stdio.h>
#include <time.h>
#include <pspecs.inc>
#include <chardef.inc>

  timup(starter)
    int  starter;
    {
      long  ktime, min;
      int   tchk, rtnval;

      time(&ktime);
      min = ktime/60;
      tchk = min % 60;

      rtnval = 0;
      if ( starter == 0 )
        {
         if ( tchk >= 5 && tchk < 8 )
            rtnval = 2;
         if ( tchk >= 35 && tchk < 38 )
            rtnval = 1;
        }

      else if ( starter == 1 )
        {
         if ( tchk >= 5 && tchk < 8 )
            rtnval = 2;
        }

      else if ( starter == 2 )
        {
         if ( tchk >= 35 && tchk < 38 )
            rtnval = 1;
        }

      ltime = ktime;
      return ( rtnval );
 }

/*===================================================================== =========
        This function checks to see if the next input record is ready.
        It first sends XON to the port, and checks for data ready.
        If not ready after a specified length of time, returns -1 to
        indicate time-out.
=========================================================================*/
```

```
chrdy(secwt,ntime)
  int  secwt;
  long *ntime;
  {
    long   outime, testim;
    int    rtnval, scwt2 = 5;

    time ( &outime);
    do
      {
        wprt ( PORTNO, CQ );          /*     Send  XON    */
        time ( ntime);
        do
          {
            if ( irdy != 0 )
              {
                rtnval = irdy - 1;
                return ( rtnval );
              }
            time (&testim);
          }
        while ( testim-*ntime < scwt2 );
      }
    while ( testim-outime < secwt );
    rtnval = -1;
    return(rtnval);
  }
```

```
          TITLE    ROUTINE TO CHECK FOR USER INTERRUPT
          NAME     funcno
;
;    THE PURPOSE OF THIS ROUTINE IS TO ALLOW A FORTRAN
;    PROGRAM TO TELL IF ANY KEY HAS BEEN STRUCK.  THE
;    ROUTINE IS ACCESSED AS AN INTEGER*2 FUNCTION, WITH
;    NO ARGUMENTS:          ITEST = IGCHR()
;    ON RETURN, THE FUNCTION VALUE IS ZERO IF NO KEY HAS
;    BEEN STRUCK.  IF A KEY HAS BEEN STRUCK, THE 8 LOW-ORDER
;    BITS OF THE FUNCTION CONTAIN THE ASCII CODE OF THE
;    CHARACTER, AND THE 8 HIGH-ORDER BITS CONTAIN THE SCAN
;    CODE. (?)  FOR THE FUNCTION KEYS, THE ASCII CODE WILL
;    BE ZERO, AND THE SCAN CODE WILL BE 3BH THROUGH 44H.
;    THE ROUTINE USES BIOS, INTERRUPT 16H TO DO ITS WORK.
;
;         PROGRAMMER:  MARY HUNT
;         DATE:        FEBRUARY 1, 1985
;
_TEXT     SEGMENT  public byte 'CODE'
          ASSUME   CS:_TEXT
;
_funcno   PROC     NEAR
PUBLIC    _funcno
;
          MOV      AH,1        ;SET AH=1 TO SEE IF A
          INT      16H        ; A CHAR. IS THERE
          JNZ      GETCHR     ; IF SF=0, THERE IS ONE
          MOV      AX,0       ; NO CHAR., SET AX = 0,
          JMP      GOHOME     ;   AND RETURN
GETCHR:   MOV      AH,0       ; THERE IS A CHAR.,
```

```
         INT      16H       ;     GO GET IT..
         cmp      al,0
         je       gfun
         mov      ax,0
         jmp      gohome
gfun:
         MOV      AL,AH
         MOV      AH,0
         SUB      AL,03AH
         cmp      al,10
         jle      gohome
         mov      al,0
;
GOHOME:  RET
_funcno  ENDP
;
_TEXT    ENDS
         END


          NAME  INTRW
; INTRW.1          March 25, 1987
;   Routine to handle interrupts from RS232 port 1,
;    port 2, or both.
;   This routine has three parts:
;    1. Initialization part, does the following:
;       a.  Sets up arguments to be used by interrupt part-
;             port number (1 or 2)
;             buffer address
;             ready flag, pointer
;             number of characters returned
;             status indicator, 0 for OK.
;       c.  Stores address of interrupt routine in interrupt vector
;       d.  Enables interrupts.
;    2. Interrupt part
;       a.  Save registers
;       b.  Get character and store in buffer
;       c.  Increment pointer
;       d.  If ASCII <E>, <*>, or buffer full, set ready flag
;           Note: LAL5000 sends ASCII EOF chars. when data block
;           complete. INTCC discards <CR>,<LF>,<NULL>,<O>,<F>
;       e.  Restore registers.
;    3. Completion part
;       a.  Inhibit interrupts
;       b.  Restore original contents of interrupt vector
;
;  ********* Original program called INTCM ***************
;
;         Programmer:  Mary M. Hunt
;         Date:        August, 1985
;
;        Modified by G. H. Power - 16 Oct., 1985
;         1. Will ignore null record, ie., LF only.
;         2. Int. routine saves cx.
;         3. Ignores null char (00H)
;*********************************************************
;=========================================================
;  Modified for medium memory model Microsoft C compiler =
;        by:  G. H. Power     Jan. 18, 1987            =
```

```
;  Calling sequence(port no.,baud code,*buffer,*ready flag,
;                  *chars. rcvd,*status,length buffer)
;  Items preceeded by <*> imply near pointers to user's
;   data area
;==========================================================
;  Modified again for small memory model Microsoft C     =
;       by:  M. M. Hunt      March 25, 1987               =
;  Also, end of record is now signalled by <CR>
;  Ignores <LF>.
;  Does not set up port; removed port no. and baud code from
;   argument list.
;  Also, sends XOFF at end of record.  Use must send XON
;   when ready.
;==========================================================
;
;    Define some constants.
;
LCR      EQU     00FBH     ;  Line Control Register
IER      EQU     00F9H     ;  Interrupt Enable Register
MCR      EQU     00FCH     ;  Modem Control Register
LSR      EQU     00FDH     ;  Line Status Register
RBR      EQU     00F8H     ;  Receiver Buffer Register
;
LINFD    EQU     0AH       ;  ASCII code for line feed
CARET    EQU     0DH       ;  ASCII code for carriage return
NULLCHR  EQU     00H       ;  ASCII code for null
ASTRISK  EQU     2AH       ;  ASCII code for <*>
ASC_B    EQU     42H       ;  ASCII code for <B>
ASC_E    EQU     45H       ;  ASCII code for <E>
ASC_O    EQU     4FH       ;  ASCII code for <O>
DOLL     EQU     24H       ;  Dollar sign
XOFF     EQU     13H       ;  ASCII code for XOFF
;
;
;       Define offsets within data area
;
DATOFF   STRUC
ADRCHB   DW         ?    ;  address of character buffer
ADRNCH   DW         ?    ;  address of number of characters
ADRPNT   DW         ?    ;             ready flag/pointer
ADRST    DW         ?    ;             status indicator
VECSV    DD         ?    ;  original contents of interrupt vector
LENSTR   DW         ?    ;  actual length of character buffer
NCHARS   DW         ?    ;  local copy of number of characters
MYFLG    DW         ?    ;  local copy of ready flag
MYST     DW         ?    ;  local copy of status indicator
PNTSTR   DW         ?    ;  string pointer
PORT     DW         ?    ;  to define port numbers (0,1)
P12INT   DB         ?    ;  to set interrupt bits
PRTIND   DB         ?    ;  use for I/O instructions
DATOFF   ENDS
;
;
;    Define code segment
_TEXT SEGMENT BYTE PUBLIC 'CODE'
_TEXT ENDS
;    Define data segment
_DATA      SEGMENT WORD PUBLIC 'DATA'
_DATA      ENDS
;
```

```
CONST      SEGMENT WORD PUBLIC 'CONST'
CONST      ENDS
;
_BSS       SEGMENT WORD PUBLIC 'BSS'
_BSS       ENDS
;
DGROUP     GROUP   CONST,  _BSS,  _DATA
;
         ASSUME CS:_TEXT ,DS: DGROUP, SS: DGROUP, ES: DGROUP
;
;   actual data storage

;          Data segment
;
_DATA        SEGMENT
         DB       'INTRW   '
         DD       _INTRW
         DD       0
;
         DB       3 DUP(0)     ; just for alignment
;
;    data area for port 1.
;
P1VAR    DB       22 DUP(?)
         DW       0
         DB       0EFH
         DB       3
;
;    data area for port 2.
;
P2VAR    DB       22 DUP(?)
         DW       1
         DB       0F7H
         DB       2
_DATA    ENDS
;
;
;          Code section
;
_TEXT  SEGMENT
;
;          Work starts here
;
         PUBLIC   _INTRW
_INTRW   PROC     NEAR
         PUSH     BP
         MOV      BP,SP
         PUSH     DI
         PUSH     SI
         PUSH     DS
         PUSH     ES
;
;    first, find port
;
         MOV      CX,[BP+4]          ; port number in CX ;
;    check port number
;
         MOV      BX,OFFSET DGROUP:P1VAR
         CMP      CX,1
         JE       DOPRT
```

```
        MOV     BX,OFFSET DGROUP:P2VAR
DOPRT:
;
;     Save addresses I will need.
;    Ready flag
;
        MOV     AX,[BP+8]
        MOV     WORD PTR ADRPNT[BX],AX
;
;    Number of characters found.
;
        MOV     AX,[BP+10]
        MOV     WORD PTR ADRNCH[BX],AX
;    Status
        MOV     AX,[BP+12]
        MOV     WORD PTR ADRST[BX],AX
;
;    now save the buffer address and length of the buffer.
;    and set up char. pointer
;
        MOV     AX,[BP+6]
        MOV     WORD PTR ADRCHB[BX],AX
        MOV     WORD PTR PNTSTR[BX],AX
        MOV     AX,[BP+14]
        MOV     WORD PTR LENSTR[BX],AX
;
;   Save DS in code segment for interrupt routine since we are
;   guaranteed nothing except CS when interrupt occurs
;
        MOV     WORD PTR CS:SAVEDS,DS
;
;   Now I think we have all addresses, etc. set up in useable form.
;   Initialize pointers, counters, etc.
;
        MOV     WORD PTR NCHARS[BX],0
        MOV     WORD PTR MYST[BX],0
        MOV     WORD PTR MYFLG[BX],1
;
;   Save address of interrupt vector for port.
;
        MOV     AH,35H
        MOV     AL,0CH
        SUB     AX,PORT[BX]
        PUSH    BX
        INT     21H
        MOV     AX,BX
        POP     BX
        MOV     WORD PTR VECSV[BX],AX
        MOV     WORD PTR VECSV[BX+2],ES
;
;   Now put address of my routine into interrupt vector.
;
        CMP     PORT[BX],0
        JE      PUT1
        MOV     DX,OFFSET CM2INT
        JMP     PUTADR
PUT1:   MOV     DX,OFFSET CM1INT
PUTADR: MOV     AL,0CH
        MOV     AH,25H
        SUB     AX,PORT[BX]
```

```
          PUSH    DS
          PUSH    CS
          POP     DS
          INT     21H
          POP     DS
;
;   Talk to the hardware -
;
          MOV     DH,PRTIND[BX]
;
          MOV     DL,MCR           ;  Modem Control Register
          MOV    AL,0BH           ;   not sure what this
          OUT     DX,AL            ;    is going to do.
;
          MOV     DL,RBR           ;  Receiver Buffer Register
          IN      AL,DX            ;   clear out the register
;
          IN      AL,21H
          AND     AL,P12INT[BX]  ; stores 0 in bit for port
          OUT     21H,AL      ; and sends to 8259 interrupt controller ;
          MOV     DL,IER      ; Interrupt Enable Register -
          MOV     AL,01H      ;  only the data available interrupt
          OUT     DX,AL
;
          POP     ES
          POP     DS
          POP     SI
          POP     DI
          MOV     SP,BP
          POP     BP
          RET                 ;  I guess that's all
;
_INTRW    ENDP
;
;   Interrupt routine
;
CM1INT    PROC    NEAR
;
;    save registers
;
          CLI
          PUSH    AX
          PUSH    DS
          PUSH    DX
          PUSH    ES
          PUSH    DI
          PUSH    SI
          PUSH    BX
          PUSH    CX
;
          MOV     DH,3
          MOV     BX,OFFSET DGROUP:P1VAR
          JMP     DOINT
;     for port 2..
;
CM2INT:
          CLI
          PUSH    AX
          PUSH    DS
          PUSH    DX
```

```
        PUSH    ES
        PUSH    DI
        PUSH    SI
        PUSH    BX
        PUSH    CX
;
        MOV     DH,2
        MOV     BX,OFFSET DGROUP:P2VAR
;
;
;   First, check for parity error
;
DOINT:
        MOV     DS,WORD PTR CS:SAVEDS   ; Restore original DS value
        JMP     DOINT1
SAVEDS: DW      0_            ; Place to tuck DS in code segment
DOINT1: MOV     DL,LSR        ; Line Status Register
        IN      AL,DX         ;  input contents
        TEST    AL,4          ;    test parity
        JZ      NOER          ; If bad,
        INC     MYST[BX]      ;   increment status indicator
;
NOER:   MOV     DL,RBR        ; Receiver Buffer Register
        IN      AL,DX         ;  Input character
        CMP     AL,CARET      ; check for <CR>
        JE      EORCOD        ;  do not store <CR>
        CMP     AL,LINFD      ; Check for line feed
        JE      RETINT        ; Do not store line feed
;
;    increment counter.
;
        MOV     CX,NCHARS[BX] ;number of characters in buffer
        INC     CX            ;    plus this one
;
;   store character in buffer
;
NEWCHR:
        MOV     NCHARS[BX],CX
        PUSH    DS
        POP     ES
        MOV     DI,PNTSTR[BX]           ; offset address
        STOSB                           ; store the byte
        MOV     PNTSTR[BX],DI           ; and update pointer
;   check if buffer full.
;
        CMP     CX,LENSTR[BX]       ; compare with buffer length
        JL      RETINT              ;   if less or +< ok.
        OR      MYST[BX],0100H      ; set flag
        JMP     EORCOD              ; end-of-record procedure
;
;   return code
;
RETINT:
        MOV     AL,64H        ;  signal end of
        SUB     AX,PORT[BX]
        OUT     20H,AL        ;   interrupt
        POP     CX
        POP     BX
        POP     SI
        POP     DI
```

```
        POP     ES
        POP     DX
        POP     DS
        POP     AX
        STI
        IRET
;
;   end-of-record procedure
;
EORCOD:
        MOV     SI,ADRNCH[BX] ;  address of no. of chars.
        MOV     AX,NCHARS[BX] ;   no. of chars. in AX
        CMP     AX,0
        JE      RETINT        ;    ignore null rec., ie LF only
        MOV     [SI],AX       ;    store no. of chars.
;
        MOV     SI,ADRST[BX]  ;  now do the same for status
        MOV     AX,MYST[BX]
        MOV     [SI],AX
;
        MOV     SI,ADRPNT[BX] ;   and for ready flag/pointer
        MOV     AX,MYFLG[BX]
        MOV     [SI],AX
;
;   Get ready for next record
;
        NEG     MYFLG[BX]     ;  next pointer will be
        ADD     MYFLG[BX],3   ;   3 - old value
;
;     Pointer
        MOV     AX,WORD PTR ADRCHB[BX] ; original value in AX
        CMP     MYFLG[BX],1            ; see which part of char. array
        JE      FLSET
        ADD     AX,LENSTR[BX]          ;  for second element
FLSET:  MOV     PNTSTR[BX],AX          ;  pointer ready for next record
;
;    reset status & no. of characters.
;
        MOV     NCHARS[BX],0
        MOV     MYST[BX],0
        MOV     AL,XOFF          ;  Send XOFF
        MOV     DL,RBR
        OUT     DX,AL
        JMP     RETINT        ;  Finished!
;
CM1INT  ENDP
;
;    Termination procedure
;
        PUBLIC  _ENRUPT
_ENRUPT PROC    NEAR
        PUSH BP
        MOV  BP,SP
        PUSH DI
        PUSH SI
        PUSH DS
        PUSH ES
        CLI
;
;  Get port number
```

```
;
        MOV     AX,[BP+4]
        MOV     BX,OFFSET DGROUP:P1VAR
        CMP     AX,1
        JE      DOEND
        MOV     BX,OFFSET DGROUP:P2VAR
;
;   Send 0 to Interrupt Enable Register
;
DOEND:
        MOV     DH,PRTIND[BX]
        MOV     DL,IER
        MOV     AX,0
        OUT     DX,AL
;   Interrupt Controller.
        IN      AL,21H
        MOV     CL,P12INT[BX]
        NOT     CL
        OR      AL,CL
        OUT     21H,AL
        STI
;
;   Restore interrupt vector
;
        MOV     AL,0CH
        MOV     AH,25H
        SUB     AX,PORT[BX]
        MOV     DX,WORD PTR VECSV[BX]
        MOV     DS,WORD PTR VECSV+2[BX]
        INT     21H
;
        POP     ES
        POP     DS
        POP     SI
        POP     DI
        MOV     SP,BP
        POP     BP
        RET
_ENRUPT  ENDP
_TEXT  ENDS
        END
; ===========================================================
;    This file has the following functions:
;       PRTSET    sets up RS232 port
;       wprt      writes one character to port
;       prnrdy    checks printer status
;============================================================
;
          NAME      PRTSET
;
;  Sets up specified port at specified baud rate.
;  Other parameters are:
;         even parity
;         7 data bits
;         1 stop bits
;  These are coded in  DSBIT, below, as follows:
;
;   bits       4  3      2            1   0
;            parity    stop bits    word length
;
```

```
;           x0 - one     0 -  1        10 - 7 bits
;           10 - odd     1 -  2        11 - 8 bits
;           11 - even
;
;  Two arguments are:
;          BAUD      baud rate
;          PORTNO    serial port number, 1 or 2
;
;            Programmer:  Mary M. Hunt
;            Date:        January, 1987
;            Originatr:   Dick Payne
;
;        Specify some constants
;
SINT        EQU         014H
THR         EQU         0F8H           ;  Transmitter Holding Register
LSR         EQU         0FDH           ;  Line Status Register
MCR         EQU         0FCH           ;  Modem Control Register
;
DGROUP      GROUP       _DATA,CONST
;
CONST       SEGMENT     WORD PUBLIC 'CONST'
;
BRATES      DW          110
            DW          150
            DW          300
            DW          600
            DW          1200
            DW         2400
            DW         4800
            DW         9600
;
NBRAT       DW         14
DSBIT       DW         01AH
;
CONST       ENDS
;
_DATA       SEGMENT    WORD PUBLIC 'DATA'
_DATA       ENDS
;
_TEXT       SEGMENT    BYTE PUBLIC 'CODE'
            ASSUME     CS:_TEXT,DS:DGROUP
            PUBLIC     _prtset
_prtset     PROC       NEAR
;
            PUSH       BP
            MOV        BP,SP
            SUB        SP,4
;
;                   get baud rate in AX
;
            MOV        AX,[BP+4]
            MOV        BX,0
LOOP:
            CMP        AX,BRATES[BX]
            JE         GOTBD
            INC        BX
            INC        BX
            CMP        BX,NBRAT
            JG         BADRTN
```

```
            JMP       LOOP
;
;                   HAVE BAUD RATE. SHIFT 5 BITS LEFT
;                   (actually only 4 bits because BX is twice
;                    as big as it should be)
;
GOTBD:
            MOV       CL,4
            SHL       BX,CL
            MOV       AX,BX
            OR        AX,DSBIT
;
            MOV       DX,[BP+6]
            SUB       DX,1
            INT       SINT
;
            MOV       AX,0
;
            MOV       DL,MCR
            MOV       DH,[BP+6]
            CMP       DH,2
            JE        OUTPT
            MOV       DH,3
OUTPT:
            MOV       AX,3
            OUT       DX,AL
RTN:
            MOV       SP,BP
            POP       BP
            RET
;
BADRTN:
            MOV       AX,1
            JMP       RTN
;
_prtset     ENDP
;
; =========================================================================
;      This function writes one character to the port.
;      The arguments are:
;           int   port      RS232 port no. (1 or 2)
;           char  c         character to be sent to port
;
;      No return value.
;
;==========================================================================
            PUBLIC    _wprt
_wprt       PROC      NEAR
;
            PUSH      BP
            MOV       BP,SP
;
;      Get port code into DH (3 for port 1, 2 for port 2)
;
            MOV       DH,[BP+4]
            CMP       DH,1
            JNE       CHKREG
            MOV       DH,3
;
CHKREG:
```

```
        MOV     DL,LSR
NOTRDY:
        IN      AL,DX           ; wait for
        TEST    AL,20H          ;   register
        JZ      NOTRDY          ;      empty
;
        MOV     DL,THR
        MOV     AX,[BP+6]       ; store character in AL
        OUT     DX,AL           ;    and output it
;
        MOV     SP,BP
        POP     BP
        RET
_wprt   ENDP
;
_TEXT   ENDS
;
;----------------------------------------------------------------------
;     Function   prnrdy
;     Checks to see if printer is ready.
;     Uses BIOS interrupt 17H.
;     Returns 0 if printer all ready, or
;     '8000'X if printer busy.  Other values mean off line.
;
;         Programmer:  Mary M. Hunt
;
;----------------------------------------------------------------------
;
_TEXT     SEGMENT     BYTE PUBLIC 'CODE'
          ASSUME      CS:_TEXT
          PUBLIC      _prnrdy
_prnrdy   PROC        NEAR
;
          PUSH        BP
          MOV         BP,SP
          MOV         AX,0200H
          MOV         DX,0
          INT         17H
;
          XOR         AX,9000H
          MOV         BP,SP
          POP         BP
          RET
;
_prnrdy   ENDP
_TEXT     ENDS
;
          END
\end
```

C. Management

C.1 FILMAN.COM Program Report

NAME:	FILMAN.COM
PURPOSE:	To manage WRPROC output files on VAX
MACHINE:	VAX
SOURCE LANGUAGE:	VAX VMS DCL

DESCRIPTION: The program WRPROC transfers two files to a VAX disk each half hour. All the files in a single UTC day bear the same name except for the version number; the latest consists of all the data for that day through the most recent half hourly transferral. The last file of the day contains all the day's data, the others are redundant. For archival purposes we organize the data into weekly files. The program is submitted to a batch queue to run at 0020 UTC (2020 EDT) each evening.

The following describes the process for summary files. The same process is used on the spectral coefficient files, except the names are WRC....DAT, not WRS....DAT. FILMAN.COM carries out the following (UTC times and days) daily:

1. Purge previous day's files to leave only last file of the day.
2. Append previous day's file to WRTHSWK.DAT.
3. Submit FILMAN.COM to batch queue to run at 0020 UTC the next day.

At 0020 UTC Saturday morning it also carries out the following:

3. Rename file WRSLSTWK.DAT to WRsyymmdd.DAT where yymmdd is the date, (870619 is 19 June 1987) of the first day of the file.
4. Rename WRSTHSWK.DAT as WRSLSTWK.DAT.
5. Open the file WRSTHSWK.DAT with column headers from file WRHEAD.DAT

INPUT:
Input is from three files: Syymmdd.DAT where yymmdd is the date WRSTHSWK.DAT and WRSLSTWK.DAT.

OUTPUT:
Output is three files or an addition to one file depending on whether it is 0020 UTC Saturday morning or not:

0020 UTC Saturday

WRSTHSWK.DAT	New file, column headers only
WRSLSTWK.DAT	Previous week's data (previous WRTHSWK.DAT)
WRSyymmdd.DATA	Previous WRSLSTWK.DAT file.

0020 UTC any other day

WRSTHSWK.DAT	with previous day's data appended.

USAGE:

FILMAN.COM submits itself to the batch queue for the next day. The log files must be checked occasionally to make sure an system error has not crashed the program before resubmission. To resubmit, type:

```
SUBMIT/NOPRINT/LOG_FILE="FILMAN.LOG"/AFTER="TODAY+20:20"

/RESTART/ - QUEUE=QUICK FILMAN.COM
```

PROGRAMMER: Richard E. Payne
ORIGINATOR: Richard E. Payne
DATE: June 1987

C.2 Program

FILMAN.COM

```
$ SET VERIFY
$ !*** Generate yesterday's date. ***
$ DAT = F$CVTIME (,"COMPARISON","DATE")
$ !*** Generate core of "S" and "C" file names ***
$ FDAT = F$EXTRACT(2,2,DAT) + F$EXTRACT(5,2,DAT)+F$EXTRACT(8,2,DAT)
$ SHOW SYMBOL FDAT
$ !*** Generate name of yesterday's "S" file ***
$ FNAMS = "S" + FDAT + ".DAT"
$ SHOW SYMBOL FNAMS
$ PURGE 'FNAMS'              !*** Purge last "S" file ***
$ !***Generate name of yesterday's "C" file
$ FNAMC = "C" + FDAT + ".DAT"
$ SHOW SYMBOL FNAMC
$ PURGE 'FNAMC'              !*** Purge last "C" file ***
$ !*** Add yesterday's "S" file to WRTHSWK.DAT ***
$ COPY WRTHSWK.DAT + 'FNAMS'   WRTHSWK.DAT
$ PURGE WRTHSWK.DAT          !*** Purge updated file
$ !*** If this is Friday, revise files ***
$ DAY = F$CVTIME ("TODAY",,"WEEKDAY")
$ SHOW SYMBOL DAY
$ IF DAY .NES. "Friday" THEN GOTO RESUB
$ DAT = F$CVTIME ("TODAY-13-00:00")
$ FDAT = F$EXTRACT(2,2,DAT)+F$EXTRACT(5,2,DAT)+F$EXTRACT(8,2,DAT)
$ SHOW SYMBOL FDAT
$ FNAMWR = "WR" + FDAT + ".DAT"
$ SHOW SYMBOL FNAMWR
$ RENAME WRLSTWK.DAT 'FNAMWR'
$ RENAME WRTHSWK.DAT WRLSTWK.DAT
$ COPY WRHEAD.DAT WRTHSWK.DAT     ! Initialize with headers
$ RESUB:   !*** Resubmit job for tomorrow
$ SUBMIT/NOPRINT/LOG_FILE="FILMAN.LOG"/AFTER="TOMORROW+20:20" -
  /RESTART/QUEUE=QUICK  FILMAN.COM
```

C.3 Log File From FILMAN.COM

```
$   set noverify                    !stop syslogin.com printing
$   exit      !end syslogin.com
$   DIR :== DIR/SIZE/DATE
$   SET TERM/VT100
%   SET-W-NOTSET, error modifying BLUE$DRC1:

-CLI-E-IVDEVTYPE, invalid device type - specify a mailbox device

$   ST132 :== SET TERM/WIDTH=132
$   ST80 :== SET TERM/WIDTH=80
$   SS132 :== SET SCREEN 132
$   SS80 :== SET SCREEN 80
$   HOME :== SET DEF PODA:[I33.REP1]
$   TELE :==SET DEF PODA:[I33.TELE]
$   SPK :== SET DEF SPAK:
$   SET VERIFY
$   !***Generate yesterday's date. ***
$   DAT = F$CVTIME (,"COMPARISON","DATE")
$   !***Generate core of "S" and "C" file names ***
$   FDAT = F$EXTRACT(2,2,DAT) + F$EXTRACT(5,2,DAT)+F$EXTRACT(8,2,DAT)
$   SHOW SYMBOL FDAT

FDAT = "870624"

$   !***Generate name of yesterday's "S" file ***
$   FNAMS = "S" + FDAT + ".DAT"
$   SHOW SYMBOL FNAMS

FNAMS = "S870624.DAT"

$   PURGE S870624.DAT               !***Purge last "S" file***
$   !***Generate name of yesterday's "C" file
$   FNAMC = "C" + FDAT + ".DAT"
$   SHOW SYMBOL FNAMC

  FNAMC = "C870624.DAT"
```

```
$  PURGE C870624.DAT          !* * * Purge last "C" file * * *
$  !* * * Add yesterday's "S" file to WRTHSWK.DAT * * *
$  COPY WRTHSWK.DAT + S870624.DAT WRTHSWK.DAT
$  PURGE WRTHSWK.DAT          !* * * Purge updated file
$  !* * * If this is Friday, revise files * * *
$  DAY = F$CVTIME ("TODAY",,"WEEKDAY")
$  SHOW SYMBOL DAY

 DAY = "Wednesday"

$  IF DAY .NES. "Friday" THEN GOTO RESUB
$  RESUB: !* * * Resubmit job for tomorrow $

SUBMIT/NOPRINT/LOG_FILE="FILMAN.LOG"/AFTER="TOMORROW+20:20"
-
   /RESTART/QUEUE=QUICK FILMAN.COM Job FILMAN (queue
RED_QUICK, entry
947) holding until 25-JUN-1987 20:20
        0.1777 C.U. USED

$  set noverify ll

 I33_REP1      job terminated at 24-JUN-1987 20:20:09.55

 Accounting information:  Buffered I/O count:  193   Peak
 working set size:   571  Direct I/O count:    270   Peak
 page file size:   2035   Page faults:         3780  Mounted

volumes:          0

 Charged CPU time:  0 00:00:02.68  Elapsed time:  0 00:00:17.64
```

D. Data Dissemination

D.1 TELECHUZ Program Report

NAME:	TELECHUZ
TYPE:	Main program
PURPOSE:	To enable unsophisticated users to access telemetered data conveniently.
MACHINE:	VAX
SOURCE LANGUAGE:	FORTRAN
DESCRIPTION:	TELECHUZ allows the user to choose data source, time period covered, and whether to display the data one screen (20 lines) at a time or to scroll through the whole file through a set of menus on the terminal screen . Appendix D.2 is a diagram of the menu structure. A present there is only one data set available, wave data from a Waverider buoy on a mooring at the Buoy Farm a few miles out of Martha's Vineyard but the menus are set up for other data sets expected to be available in 1988. It is eminently expandable can something be done about the May in here.

INPUT:

Input is from files on any disk drive on the WHOI DECNET. A naming convention must be adopted so that TELECHUZ can determine appropriate file names at any time. The user does, not have to know the filename, as TELECHUZ automatically accesses the connect file, according to the menu choice.

For the Waverider data, TELECHUZ can access one of three summary files:

SYYMMDD.DAT	The current day's data, i.e., that acquired since 0000UTC of the current day. YY is year (last two digits). MM and DD are month and day. Example is S870610.DAT for 10 June 1987.
WRSTHSWK.dat	The current weeks's data or the data accumulated since 0000UTC on the previous Saturday.
WRLSTWK.DAT	The previous week's data, starting at 0000UTC on the previous Saturday.

The spectral coefficient files can also be accesed with TELECHUZ. In the naming convention, the S for summary is replaced by C for coefficient.

OUTPUT:

The program lists data by the screen or by the file, user's menu option. As presently configured for the Waverider data, the user can get 20 lines at a time with the program waiting for a <CR> before sending the next 20 lines, or he can have the whole file sent with no interruptions. The latter option is included for users who wish to capture the file on a PC. Both options have column headers. An additional option is for a screen which describes the parameters in the data files. The last parameter in each Waverider record, the frequency of the spectral peak, is converted to a period before being transmitted to the terminal.

USAGE:

Dialing the WHOI Red VAX (from inside the Institution, extension 6800 for 300/1200 baud Vadic modems, or extension 6815 for 300/1200 baud Bell modems) connects the user to the Institution PACX. From a telephone outside the Institution, (617) 540-6000 is a direct line to the PACX. Typing <CR> gets the PACX prompt. Appendix D.3 is an example of a TELECHUZ session which will illustrate the initial commands required by the Institution computer system. The password for this account can be obtained from any of the authors of this report. Note that the Password is not echoed to the user and does not appear on his screen. Menus and prompts will lead the user the program.

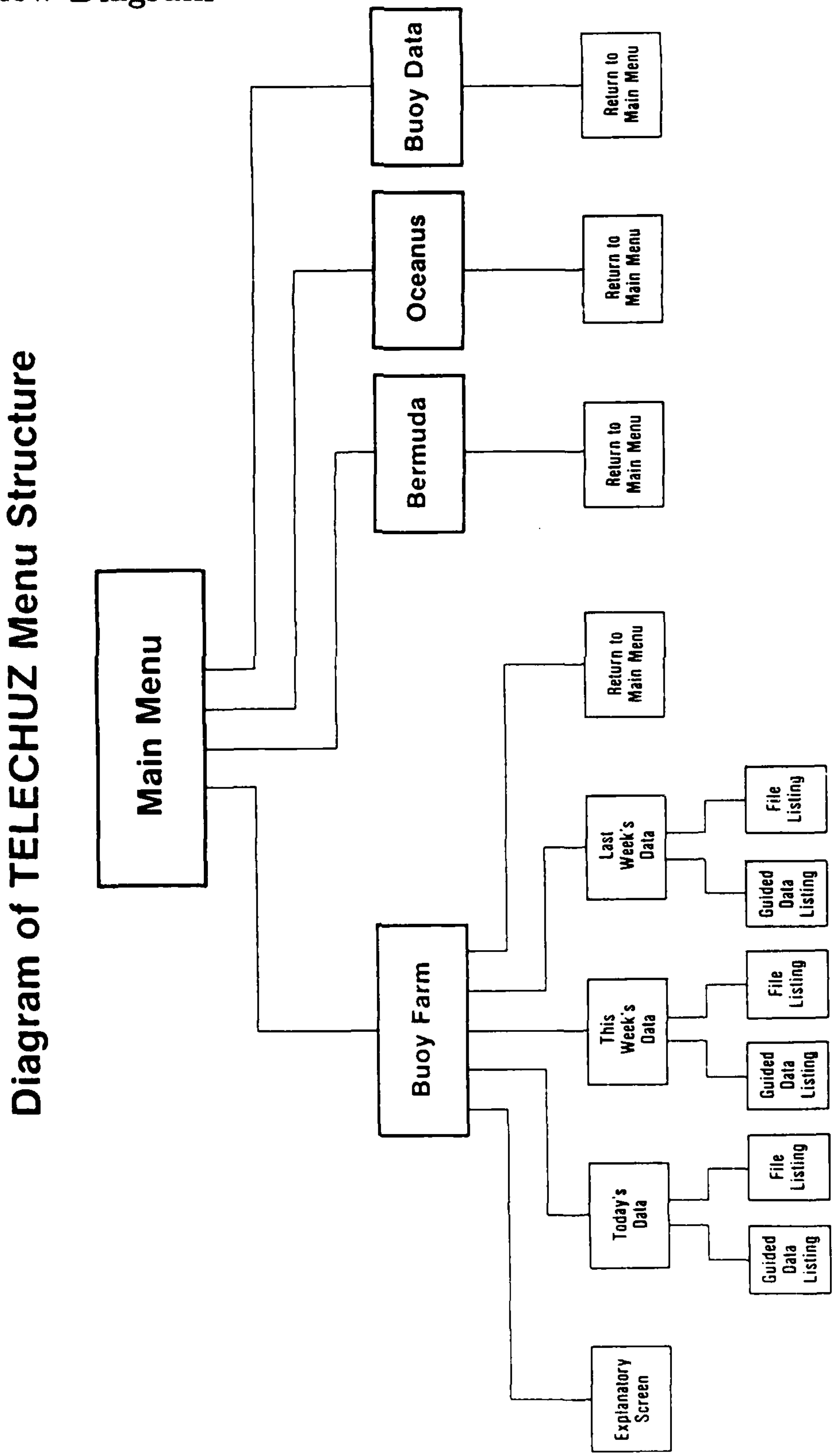
For security purposes, the VAX account has been set up with CTRL Y, CTRL C disabled. A logon file in the account puts the user immediately into the program. He cannot leave the program without logging himself off. With these safeguards the user has access only to files available through TELECHUZ.

SUBPROGRAMS USED:

File	Description
BUOYFARM.FOR	Provides menu for three categories of Waverider data, current day, current week, previous week.
WRDAT.FOR	Provides access to the Waverider files.
BERMUDA.FOR	Dummy subroutine until data are available.
OCEANUS.FOR	Dummy subroutine until data are available.
BUOYDATA.FOR	Dummy subroutine until data are available.

PROGRAMMER: Richard E. Payne
ORIGINATOR: Melbourne Briscoe
DATE: June 1987

D.1a Flow Diagram



D.2 Sample TELECHUZ Session

```
enter class: RED
class red start

Username: I33_TELE
Password:

WHOI RED system

    Last interactive login on Friday, 12-JUN-1987 15:06

************************************************************

                   WELCOME TO
               URIP TELEMETERED DATA
       You will be guided through the latest
       telemetered data by a system of menus

       Please type your name
REP

************************************************************

                   FIRST LEVEL MENU
       1. Buoy Farm Wave Rider data (start May 1987)
       2. Bermuda data (not yet available)
       3. Oceanus data (not yet available)
       4. Buoy data (not yet available)
       5. Logoff

    Please enter desired item number followed by a <CR>...
1
************************************************************

       DATA CATEGORIES FOR BUOY FARM WAVE RIDER DATA

       0. Format explanation
       1. Current 24 Hours
       2. Current week
       3. Previous week
       4. Return to previous menu

    For earlier data, contact R.Payne at 617-548-1400 ext 2550.
    Please type desired item number followed by a <CR>...
0
WAVERIDER DATA
```

```
UTC TIME-In hours, minutes. Subtract 4 hours for EDT
VARIANCE-Total variance of sea height after removing mean, in m**2
         zeroth moment of spectrum, M0
SIG WAVE HEIGHT-4*SQRT(VARIANCE), in meters
MEAN WAVE PERIOD-M0/M1, in seconds
UPCROSS WAVE PERIOD-Zero upcrossing period=, SQRT(M0/M2), in seconds
WAVERIDER BIAS-Mean of all 2048 sea heights in data set, in meters
MAX CREST-Maximum sea height in data set, mean removed, in meters
MIN TROUGH-Minimum sea height in data set, mean removed, in meters
SPECTRAL PEAK PERIOD-Period of largest spectral component, in seconds

Type <CR> to return to menu

************************************************************

       DATA CATEGORIES FOR BUOY FARM WAVE RIDER DATA

       0. Format explanation
       1. Current 24 Hours
       2. Current week
       3. Previous week
       4. Return to previous menu

    For earlier data, contact R.Payne at 617-548-1400 ext 2550.
    Please type desired item number followed by a <CR>...
2
For prompted listing of file, type "1"<CR>
For uninterrupted listing (dump to PC), type "2"<CR>
File ID for you to copy from to another account is
           PODA:[I33.REP1]WRTHSWK.DAT
1
************************************************************
```

For first 20 lines of data, type <CR>. Type "E"<CR> to return to previous menu.

DATE	UTC TIME	VAR-IANCE	SIG WAVE HEIGHT	MEAN WAVE PERIOD	UPCROSS WAVE PERIOD	WAVE RIDER BIAS	MAX CREST	MIN TROUGH	SPECTRAL PEAK PERIOD
870606	0	0.030	0.690	6.039	5.386	0.411	0.910	-1.011	7.09
870606	30	0.037	0.771	5.701	4.913	0.412	1.353	-1.711	7.52
870606	100	0.036	0.763	5.698	4.873	0.360	0.967	-1.208	6.76
870606	130	0.043	0.825	5.600	4.866	0.401	0.995	-1.149	7.52
870606	200	0.049	0.883	4.711	4.099	0.371	1.239	-1.839	7.52
870606	230	0.036	0.756	4.866	4.257	0.402	1.175	-1.232	7.09
870606	300	0.041	0.805	5.363	4.584	0.398	1.104	-1.437	6.41

```
     870606   330  0.031  0.701  4.892  4.227  0.404  1.026  -1.380   6.41
     870606   400  0.027  0.663  5.117  4.549  0.396  0.601  -0.571   5.81
     870606   430  0.031  0.705  5.241  4.598  0.391  0.809  -0.647   8.00
     870606   500  0.029  0.685  4.882  4.332  0.395  0.693  -0.683   7.09
     870606   530  0.028  0.667  4.638  4.066  0.396  0.795  -0.645   6.76
     870606   600  0.028  0.668  4.496  3.939  0.397  0.709  -0.566   6.10
     870606   630  0.029  0.687  4.834  4.176  0.396  0.703  -0.887   7.09
     870606   700  0.028  0.666  4.561  3.972  0.391  0.814  -0.736   6.10
     870606   730  0.025  0.635  4.368  3.883  0.393  0.656  -0.619   5.81
     870606   800  0.027  0.653  4.682  4.132  0.386  0.680  -0.658   6.41
     870606   830  0.023  0.613  4.785  4.215  0.391  0.765  -0.584   7.09
     870606   900  0.021  0.579  4.720  4.170  0.387  0.669  -0.580   7.09
     870606   930  0.020  0.563  4.805  4.236  0.387  0.672  -0.534   8.55
     Type <CR> for next 20 lines of data, E <CR> to return to previous
menu.
     E
     Returning you to previous menu.

     ************************************************************

            DATA CATEGORIES FOR BUOY FARM WAVE RIDER DATA

            0. Format explanation
            1. Current 24 Hours
            2. Current week
            3. Previous week
            4. Return to previous menu

         For earlier data, contact R.Payne at 617-548-1400 ext 2550.
         Please type desired item number followed by a <CR>...
     4

     ************************************************************

                FIRST LEVEL MENU
            1. Buoy Farm Wave Rider data (start May 1987)
            2. Bermuda data (not yet available)
            3. Oceanus data (not yet available)
            4. Buoy data (not yet available)
            5. Logoff

         Please enter desired item number followed by a <CR>...
     5
     Please wait for the 2 line VAX logoff message before hanging up.
Thank you.
               0.1477 C.U. USED
       I33_TELE     finished at 12-JUN-1987 16:10:29.42
```

D.3 Program

```
C*******************************************************************
C TELECHUZ.FOR
C R.E.Payne   13 February 1987
C*******************************************************************
C TELECHUZ provides access to telemetry data from an account on the
C Red VAX. It is expandable through subroutines to allow any number
C of channels and levels.
C   23 Jun 87 - Added exit from any menu.
C*******************************************************************
C LINK TELECHUZ,BUOYFARM,WRDAT,BERMUDA,OCEANUS,BUOYDATA
C*******************************************************************
C***Type specification statements***
      INTEGER*2 IEND
      CHARACTER CHUZ*12,NAME*32,DMY*9,HMS*8
C***Write initial screen***
      WRITE (6,1000)
C***Write first level menu***
  100 CONTINUE
      READ (5,1020) NAME
      OPEN (UNIT=1,FILE='[I33.REP1]USERS.DOC',STATUS='OLD',
     &   ACCESS='APPEND')
      CALL DATE (DMY)
      CALL TIME (HMS)
      WRITE (1,1050) NAME,HMS,DMY
      CLOSE (UNIT=1,STATUS='KEEP')
  150 CONTINUE
      IEND = 0
      WRITE (6,1100)
  200 CONTINUE
      READ (5,1150) CHUZ
C***CHUZ is channel choice on first level***
      IF (CHUZ.EQ.'1') THEN
         CALL BUOYFARM (IEND)
      ELSEIF (CHUZ.EQ.'2') THEN
         CALL BERMUDA (IEND)
      ELSEIF (CHUZ.EQ.'3') THEN
         CALL OCEANUS (IEND)
      ELSEIF (CHUZ.EQ.'4') THEN
         CALL BUOYDATA (IEND)
      ELSEIF (CHUZ.EQ.'5') THEN
         IEND = 1
      ELSE
         WRITE (6,1200) CHUZ
         GOTO 200
      ENDIF
```

```
      IF (IEND.EQ.0) THEN
         GOTO 150
      ENDIF
C***Exeunt***
  998 CONTINUE
      WRITE (6,1300)
C***Format statments***
 1000 FORMAT (/,' ************************************************',
     &   '*************',//,
     &   20X,'WELCOME TO',/,
     &   16X,'URIP TELEMETERED DATA',/,
     &   8X,'You will be guided through the latest',/,
     &   8X,'telemetered data by a system of menus',//,
     &   8X,'Please type your name')
 1020 FORMAT (A32)
 1050 FORMAT (4X,A32,2X,A9,2X,A8)
 1100 FORMAT (/,' ************************************************',
     &   '************* ',//,
     &   20X,'FIRST LEVEL MENU',/,
     &   8X,'1. Buoy Farm Wave Rider data (start May 1987)',/,
     &   8x,'2. Bermuda data (not yet available)',/,
     &   8x,'3. Oceanus data (not yet available)',/,
     &   8x,'4. Buoy data (not yet available)',/,
     &   8x,'5. Logoff',//,
     &   5x,'Please enter desired item number followed ',
     &      'by a <CR>...  ')
 1150 FORMAT (A12)
 1200 FORMAT (' I understood you to type ',A12,'. Please retype ',
     &   'choice, "1" to "5"')
 1300 FORMAT (' Please wait for the 2 line VAX lo~off message',
     &   ' before hanging up. Thank you.')
      END

C*******************************************************************
      SUBROUTINE BUOYFARM (IEND)
C     R.E.Payne   13 February 1987
C*******************************************************************
C BUOYFARM contains a menu for choosing to access the three
C categories of Buoy Farm Wave Rider data, current day, current
C week, previous week.
C MODIFICATIONS:
C    7 May 87 - Add computed file name for today's data.
C   12 Jun 87 - Added 0 category to menu to explain data.
C*******************************************************************
C***Type specification statements***
      INTEGER*2 Y,M,D,IEND
      REAL*4 RDATE
```

```
      CHARACTER CHUZ*12,INFILE*26,S*1,SUF*3
C***Initialization***
      S = 'S'
      SUF = 'DAT'
      IEND = 0
C***Write menu screen***
C   0 - Print explanation screen
C   1 - Access today's data file
C   2 - Access WRTHSWK.DAT
C   3 - Access WRLSTWK.DAT
C   4 - Return to previous screen
C   5 - Exit program
  100 CONTINUE
      WRITE (6,1000)
  200 CONTINUE
      READ (5,1050) CHUZ
      IF (CHUZ.EQ.'0') THEN
         WRITE (6,1075)
         READ (5,1050)
         GOTO 100
      ELSEIF (CHUZ.EQ.'1') THEN
         CALL IDATE (M,D,Y)
         RDATE = REAL(D) + 100.*REAL(M) + 10000.*REAL(Y)
         WRITE (INFILE,1200) S,RDATE,SUF
         INFILE = 'PODA:[I33.REP1]'//INFILE
         CALL WRDAT (INFILE)
      ELSEIF (CHUZ.EQ.'2') THEN
         INFILE = 'PODA:[I33.REP1]WRTHSWK.DAT'
         CALL WRDAT (INFILE)
      ELSEIF (CHUZ.EQ.'3') THEN
         INFILE = 'PODA:[I33.REP1]WRLSTWK.DAT'
         CALL WRDAT (INFILE)
      ELSEIF (CHUZ.EQ.'4') THEN
         RETURN
      ELSEIF (CHUZ.EQ.'5') THEN
         IEND = 1
         RETURN
      ELSE
         WRITE (6,1100) CHUZ
         GOTO 200
      ENDIF
      GOTO 100

C***Format statements***
 1000 FORMAT (' ***************************************************',
     &    '***********',//,
     &    8X,'DATA CATEGORIES FOR BUOY FARM WAVE RIDER DATA',//,
```

```
     &   8X,'0. Format explanation',/,
     &   8X,'1. Current 24 Hours',/,
     &   8X,'2. Current week',/,
     &   8X,'3. Previous week',/,
     &   8X,'4. Return to previous menu',/,
     &   8X,'5. Exit program.',//,
     &   5X,'For earlier data, contact R.Payne at 617-548-1400',
     &   ' ext 2550.',/,
     &   5X,'Please type desired item number followed ',
     &      'by a <CR>...  ')
 1050 FORMAT (A1)
 1075 FORMAT (' WAVERIDER DATA',/,
     &   ' UTC TIME-In hours, minutes. Subtract 4 hours for EDT',/,
     &   ' VARIANCE-Total variance of sea height after removing',
     &      ' mean, in m**2',/,
     &   '            zeroth moment of spectrum, M0',/,
     &   ' SIG WAVE HEIGHT-4*SQRT(VARIANCE), in meters',/,
     &   ' MEAN WAVE PERIOD-M0/M1, in seconds',/,
     &   ' UPCROSS WAVE PERIOD-Zero upcrossing period =',
     &      ' SQRT(M0/M2), in seconds',/,
     &   ' WAVERIDER BIAS-Mean of all 2048 sea heights in data',
     &      ' set, in meters',/,
     &   ' MAX CREST-Maximum sea height in data set, mean',
     &      ' removed, in meters',/,
     &   ' MIN TROUGH-Minimum sea height in data set, mean',
     &      ' removed, in meters',/,
     &   ' SPECTRAL PEAK PERIOD-Period of largest spectral',
     &      ' component, in seconds',/,
     &   ' ',/,
     &   ' Type <CR> to return to menu')
 1100 FORMAT (' I understood you to type ',A12,'. Please retype ',
     &   'choice, "1" to "4"')
 1200 FORMAT (A1,F7.0,A3)
      END

C*****************************************************************
      SUBROUTINE WRDAT (INFILE)
C     R.E.Payne   17 February 1987
C*****************************************************************
C WRDAT displays Wave Rider data file.
C MODIFICATIONS:
C    7 May 87 - Changed to format of actual data.
C   20 May 87 - Changed column headers
C    8 Jun 87 - Added discard of input file headers
C   12 Jun 87 - Added inversion of last parameter in each record
C   23 Jun 87 - Changed termination at end of data file.
C*****************************************************************
```

```
C***Type specification statements
         INTEGER*2 NUMHEAD,IERR
         INTEGER*4 IV1(25),IV2(25)
         REAL*4 RV1(25),RV2(25),RV3(25),RV4(25),RV5(25),
      &   RV6(25),RV7(25),RV8(25)
         CHARACTER INFILE*26,CHUZ*12,CR*1
         CHARACTER*80 HEADER(3),REC
C***Initialization***
         OPEN (UNIT=1,FILE=INFILE,STATUS='OLD',ERR=500,
      &   IOSTAT=IERR,READONLY)
C   Read and discard header records for this or last week's file
         IF (INFILE.EQ.'PODA:[I33.REP1]WRTHSWK.DAT') THEN
          READ (1,1000)(HEADER(I),I=1,3)
         ELSEIF (INFILE.EQ.'PODA:[I33.REP1]WRLSTWK.DAT') THEN
          READ (1,1000)(HEADER(I),I=1,3)
         ENDIF
C***Inquire for file transfer or conducted listing***
         WRITE (6,1200) INFILE
     50 CONTINUE
         READ (5,1300) CHUZ
         CALL STR$UPCASE (CHUZ,CHUZ)
C****************************************************************
C***Test for output mode***
C***Guided listing of data file***
         IF (CHUZ.EQ.'1') THEN
          WRITE (6,2050)
          WRITE (6,2000)
          READ (5,2300) CR
          CALL STR$UPCASE (CR,CR)
          IF (CR.EQ.'E') THEN
           CLOSE (UNIT=1,STATUS='KEEP')
           RETURN
          ENDIF
C   Read 20 data records and invert last variable
  110    CONTINUE
          N = 0
          DO 120 I=1,20
          READ (1,*,END=130) IV1(I),IV2(I),RV1(I),RV2(I),
     &         RV3(I),RV4(I),RV5(I),RV6(I),RV7(I),RV8(I)
          RV8(I) = 1./RV8(I)
          N = I
  120    CONTINUE
  130    CONTINUE
C***Test if this is end of data
C   If no records were read, terminate
         IF (N.EQ.0) THEN
          WRITE (6,2100)
```

```
             CLOSE (UNIT=1,STATUS='KEEP')
             RETURN
            ENDIF
C   Otherwise, write N records with header
            WRITE (6,2600)
            WRITE (6,1100)(IV1(I),IV2(I),RV1(I),RV2(I),RV3(I),
     &         RV4(I),RV5(I),RV6(I),RV7(I),RV8(I),I=1,N)
C   If N < 20, write message, terminate
            IF (N.LT.20) THEN
             WRITE (6,2100)
             READ (5,2300) CR
             CLOSE (UNIT=1,STATUS='KEEP')
             RETURN
C   If N = 20, prompt for further action
            ELSEIF (N.EQ.20) THEN
C   Prompt for more data
             WRITE (6,2200)
             READ (5,2300) CR
             CALL STR$UPCASE (CR,CR)
C   If space bar, then read next N lines
             IF (CR.EQ.'E') THEN
              WRITE (6,2400)
              CLOSE (UNIT=1,STATUS='KEEP')
              RETURN
             ELSE
              GOTO 110
             ENDIF
            ENDIF
C****************************************************************
C***File transfer to terminal***
         ELSEIF (CHUZ.EQ.'2') THEN
            WRITE (6,2600)
            DO 200 I=1,1000
             READ (1,*,END=210) IV1(1),IV2(1),RV1(1),RV2(1),RV3(1),
     &            RV4(1),RV5(1),RV6(1),RV7(1),RV8(1)
             RV8(1) = 1./RV8(1)
             WRITE (6,1100) IV1(1),IV2(1),RV1(1),RV2(1),RV3(1),
     &            RV4(1),RV5(1),RV6(1),RV7(1),RV8(1)
  200       CONTINUE
  210       CONTINUE
            WRITE (6,2100)
            READ (5,2300) CR
            CLOSE (UNIT=1,STATUS='KEEP')
            RETURN
C****************************************************************
         ELSE
            WRITE (6,1400) CHUZ
```

```
          GOTO 50
       ENDIF
C***Error in OPEN of input file***
  500 CONTINUE
       WRITE (6,2500) IERR
       RETURN
C***Format statements***
 1000 FORMAT (A77)
 1100 FORMAT (2X,I6,2X,I4,2X,F5.3,2X,F5.3,2X,F5.3,2X,F5.3,
     &   2X,F5.3,2X,F5.3,2X,F6.3,2X,F5.2)
 1200 FORMAT (' For prompted listing of file, type "1"<CR>',/,
     &   ' For uninterrupted listing (dump to PC), type "2"<CR>',/,
     &   ' File ID for you to copy from to another account is ',/,
     &   6X,A32)
 1300 FORMAT (A12)
 1400 FORMAT (' I understood you to type ',A12,
     &   'Please retype choice, "1" or "2"')
 2000 FORMAT (/,' For first 20 lines of data, type <CR>.',
     &   ' Type "E"<CR> to return to previous menu. ')
 2050 FORMAT (' **********************************************',
     &   '***********',/)
 2100 FORMAT (' End of data, type <CR> to return',
     &   ' to previous menu.')
 2200 FORMAT (' For next 20 lines of data, type <CR>.',
     &   ' Type E <CR> to return to previous menu. ')
 2300 FORMAT (A1)
 2400 FORMAT (' Returning you to previous menu.',/)
 2500 FORMAT (' No data in that file. Returning you to',
     &   ' previous menu. IERR=',I5,/)
 2600 FORMAT ( 17X,'       SIG   MEAN  UPCROSS  WAVE',
     &   '                SPECTRAL',/,
     &         11X,'UTC  VAR-   WAVE   WAVE   WAVE   RIDER',
     &   '  MAX     MIN    PEAK',/,
     & '  DATE   TIME  IANCE HEIGHT PERIOD PERIOD   BIAS',
     &   '  CREST  TROUGH  PERIOD')
       END

C*******************************************************************
       SUBROUTINE BERMUDA (IEND)
C      R.E.Payne   13 February 1987
C*******************************************************************
C BERMUDA contains a menu for choosing to access the categories of
C Bermuda data.
C*******************************************************************
C***Type specification statements***
       INTEGER*2 CHUZ,IEND
C***Initialization***
```

```
      IEND = 0
C***Write comment***
      WRITE (6,1000)
      RETURN
C***Format statements***
 1000 FORMAT (' There is no Bermuda data yet. We expect it to ',
     &   'start on about June ?',/,
     &   ' You will now be returned to the previous menu.')
      END

C*****************************************************************
      SUBROUTINE OCEANUS (IEND)
C     R.E.Payne   13 February 1987
C*****************************************************************
C OCEANUS contains a menu for choosing to access the categories of
C Oceanus data.
C*****************************************************************
C***Type specification statements***
      INTEGER*2 CHUZ,IEND
C***Initialization***
      IEND = 0
C***Write comment***
      WRITE (6,1000)
      RETURN
C***Format statements***
 1000 FORMAT (' There is no Oceanus data yet. We expect it to ',
     &   'start on about June ?',/,
     &   ' You will now be returned to the previous menu.')
      END

C*****************************************************************
      SUBROUTINE BUOYDATA (IEND)
C     R.E.Payne   13 February 1987
C*****************************************************************
C BUOYDATA contains a menu for choosing to access the categories of
C BUOYDATA data.
C*****************************************************************
C***Type specification statements***
      INTEGER*2 CHUZ,IEND
C***Initialization***
      IEND = 0
C***Write comment***
      WRITE (6,1000)
      RETURN
C***Format statements***
 1000 FORMAT (' There is no buoy data yet. We expect it to start',
     &   ' on about October ?',/,
```

```
&    ' You will now be returned to the previous menu.')
 END
```

DOCUMENT LIBRARY

August 21, 1987

Distribution List for Technical Report Exchange

Attn: Stella Sanchez-Wade
Documents Section
Scripps Institution of Oceanography
Library, Mail Code C-075C
La Jolla, CA 92093

Hancock Library of Biology &
Oceanography
Alan Hancock Laboratory
University of Southern California
University Park
Los Angeles, CA 90089-0371

Gifts & Exchanges
Library
Bedford Institute of Oceanography
P.O. Box 1006
Dartmouth, NS, B2Y 4A2, CANADA

Office of the International
Ice Patrol
c/o Coast Guard R & D Center
Avery Point
Groton, CT 06340

Library
Physical Oceanographic Laboratory
Nova University
8000 N. Ocean Drive
Dania, FL 33304

NOAA/EDIS Miami Library Center
4301 Rickenbacker Causeway
Miami, FL 33149

Library
Skidaway Institute of Oceanography
P.O. Box 13687
Savannah, GA 31416

Institute of Geophysics
University of Hawaii
Library Room 252
2525 Correa Road
Honolulu, HI 96822

Library
Chesapeake Bay Institute
4800 Atwell Road
Shady Side, MD 20876

MIT Libraries
Serial Journal Room 14E-210
Cambridge, MA 02139

Director, Ralph M. Parsons Laboratory
Room 48-311
MIT
Cambridge, MA 02139

Marine Resources Information Center
Building E38-320
MIT
Cambridge, MA 02139

Library
Lamont-Doherty Geological
Observatory
Colombia University
Palisades, NY 10964

Library
Serials Department
Oregon State University
Corvallis, OR 97331

Pell Marine Science Library
University of Rhode Island
Narragansett Bay Campus
Narragansett, RI 02882

Working Collection
Texas A&M University
Dept. of Oceanography
College Station, TX 77843

Library
Virginia Institute of Marine Science
Gloucester Point, VA 23062

Fisheries-Oceanography Library
151 Oceanography Teaching Bldg.
University of Washington
Seattle, WA 98195

Library
R.S.M.A.S.
University of Miami
4600 Rickenbacker Causeway
Miami, FL 33149

Maury Oceanographic Library
Naval Oceanographic Office
Bay St. Louis
NSTL, MS 39522-5001

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. WHOI-88-15	2.	3. Recipient's Accession No.
4. Title and Subtitle **Surface-Wave Data Acquisition and Dissemination by VHF Packet Radio and Computer Networking**			5. Report Date April 1988 6.
7. Author(s) M. Briscoe, E. Denton, D. Frye, M. Hunt, E. Montgomery, and R. Payne			8. Performing Organization Rept. No. WHOI-88-15
9. Performing Organization Name and Address The Woods Hole Oceanographic Institution Woods Hole, Massachusetts 02543			10. Project/Task/Work Unit No. 11. Contract(C) or Grant(G) No. (C) N00014-86-K-0751 (G)
12. Sponsoring Organization Name and Address The Office of Naval Research Environmental Sciences Directorate Arlington, Virginia 22217			13. Type of Report & Period Covered Technical Report 14.

15. Supplementary Notes

This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept., WHOI-88-15.

16. Abstract (Limit: 200 words)

Waverider buoy data are normally transmitted on a 27 MHz analog radio link to a shore station a few miles away, where the buoy data are plotted on a paper strip-chart recorder or logged digitally for later computer processing.

Instead, we constructed a relay station on Martha's Vineyard island that retransmits the received Waverider data over a digital, 148 MHz packet-radio link to a personal computer in our laboratory on Cape Cod, where the data are edited, processed, spectrally analyzed, and then sent over an Ethernet line to our Institution mainframe computer for archiving. Telephone modem access of a special wave-data file on the mainframe permits unattended data dissemination to the public.

The report describes the entire system including Waverider buoy mooring hardware, computer programs, and equipment.

The purpose of the project was to learn what difficulties are involved in the automated acquisition and dissemination of telemetered oceanographic data, and to gain experience with packet radio techniques. Although secondary to these purposes, the long-term surface-wave monitoring off the southwest shore of Martha's Vineyard has its own scientific, engineering, and environmental benefits.

17. Document Analysis a. Descriptors

1. Radio frequency data telemetry
2. Packet radio
3. Automated acquisition and dissemination

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Approved for publication; distribution unlimited.	UNCLASSIFIED	113
	20. Security Class (This Page)	22. Price

(See ANSI-Z39.18) *See Instructions on Reverse* OPTIONAL FORM 272 (4-77) (Formerly NTIS-35) Department of Commerce

END
DATE
FILMED
DTIC
9-88